НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «*Сервіс інтеграції програмного забезпечення*»

Виконала: студентка *IV* курсу, групи *ІС-з61*
Щербакова Анастасія Дмитрівна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *ст. вик. Олійник Юрій Олександрович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *к.т.н, доцент Тєлишева Тамара Олексіївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *доц. каф ТК, к.т.н., доц. Ткач М.М.* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2020 року

UNICHECK

Власник документу:
Попенко Володимир Дмитрович
Дата перевірки:
11.06.2020 01:34:50 EEST
Дата звіту:
12.06.2020 01:15:44 EEST

ID перевірки:
1003947934
Тип перевірки:
Doc vs Internet + Library
ID користувача:
77149

Назва документу: Shcherbakova_isz61_2.doc

ID файлу: 1003963085 Кількість сторінок: 40 Кількість слів: 4366 Кількість символів: 32375 Розмір файлу: 1.36 MB

10.2% Схожість

Найбільша схожість: 3.16% з джерело бібліотеки. ID файлу: 1000017950

4.7% Схожість з Інтернет джерелами 24 ..... Page 42

8.93% Текстові збіги по Бібліотеці акаунту 147 ..... Page 42

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

Підміна символів

Не знайдено замінених символів

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) інформатики та обчислювальної техніки
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2020 р.

З А В Д А Н Н Я
на дипломний проєкт студенту

Щербаковій Анастасії Дмитрівні
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Сервіс інтеграції програмного забезпечення*»

керівник проєкту *Олійник Юрій Олександрович, ст. вик.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*7*”*травня* 2020 р. №1089-с

2. Термін подання студентом проєкту “*01*”*червня* 2020 року

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема бази даних

2. Схема структурна послідовності

3. Схема структурна класів програмного забезпечення

4. Схема структурна компонентів програмного забезпечення

5. Схема структурна варіантів використань

6. Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*01.02.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*10.02.2020*	
3.	*Постановка та формалізація задачі*	*15.02.2020*	
4.	*Розробка інформаційного забезпечення*	*20.03.2020*	
5.	*Алгоритмізація задачі*	*25.03.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*27.03.2020*	
7.	*Розробка програмного забезпечення*	*24.04.2020*	
8.	*Налагодження програми*	*30.04.2020*	
9.	*Виконання графічних документів*	*02.05.2020*	
10.	*Оформлення пояснювальної записки*	*10.05.2020*	
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Анастасія ЩЕРБАКОВА

Керівник Юрій ОЛІЙНИК

№з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6229.00.000 ПЗ	Пояснювальна записка	67	
3	А4	ДП 6229.01.000 ТЗ	Технічне завдання	10	
4	А3	ДП 6229.02.000 СБД	Схема бази даних	1	
5	А3	ДП 6229.03.000 ССП	Схема структурна послідовності	1	
6	А3	ДП 6229.04.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
7	А3	ДП 6229.05.000 ССК	Схема структурна компонентів	1	
			програмного забезпечення		
8	А3	ДП 6229.06.000 ССВ	Схема структурна варіантів використань	1	
9	А3	ДП 6229.07.000 КЕ	Креслення вигляду екранних форм	1	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6229.1089-с		
Розробн.		Щербакова А.Д.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Олійник Ю.О.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-з61	
Н/контр.		Тєлишева Т.О.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Сервіс інтеграції програмного забезпечення

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з шести розділів, містить 30 рисунків, 11 таблиць, 1 додаток, 5 джерел.

Дипломний проєкт присвячений розробці сервісу інтеграцій програмного забезпечення. Метою розробки є збільшення можливостей інтеграції програмного забезпечення завдяки створенню динамічних типів та систем, конфігурування точок доступу, налаштовуванню відношень між типами програмного забезпечення.

У розділі «Інформаційне забезпечення» описана структура вхідних та вихідних даних, наведена структура бази даних.

У розділі «Математичне забезпечення» наведено детальний опис математичних алгоритмів, що були використані.

У розділі «Програмне та технічне забезпечення» описані основні програмні засоби, які були використані для розробки даної програми, наведені технічні вимоги до системи, на якій буде запускатися програма, описана програмна архітектура, яка була обрана для розробки.

ІНТЕГРАЦІЇ, ДИНАМІЧНА ТИПІЗАЦІЯ, ШИНА ДАНИХ, ТРАНСФОРМУВАННЯ ДАНИХ

ABSTRACT

Structure and scope of work. Explanatory note of the diploma project consists of six sections, containing 30 figures, 11 tables, 1 supplement, 5 sources.

The diploma project is devoted to the development of a software integration service. The purpose of the development is to increase the possibilities of software integration by creating dynamic types and systems, configuring access points, setting up relationships between types of software.

The section "Information support" describes the structure of input and output data, the structure of the database.

The section "Mathematical software" provides a detailed description of the mathematical algorithms used.

The section "Software and hardware" describes the main software tools that were used to develop this program, provides technical requirements for the system on which to run the program, describes the software architecture that was selected for development.

INTEGRATION, DYNAMIC TYPING, DATA BUS, DATA TRANSFORMATION

ЗМІСТ

ВСТУП **6**

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ **7**

1.1 Опис предметного середовища 7

1.1.1 Опис процесу діяльності *7*

1.1.2 Опис функціональної моделі *8*

1.2 Огляд наявних аналогів 10

1.3 Постановка задачі 11

1.3.1 Призначення розробки *11*

1.3.2 Цілі та задачі розробки *12*

Висновок до розділу

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ **13**

2.1 Вхідні дані 13

2.2 Вихідні дані 13

2.3 Опис структури бази даних 14

Висновок до розділу 23

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ **24**

3.1 Змістовна постановка задачі 24

3.2 Математична постановка задачі 24

3.3 Обґрунтування методу розв'язання 27

3.4 Опис методу розв'язання 28

Висновок до розділу 28

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ **29**

4.1 Засоби розробки 29

4.2 Вимоги до технічного забезпечення 30

4.2.1 Загальні вимоги *30*

4.3 Архітектура програмного забезпечення 30

4.3.1 Діаграма класів *30*

4.3.2 Діаграма послідовності *31*

4.3.3 Діаграма компонентів *31*

4.3.4 Специфікація функцій 32

ВИСНОВОК ДО РОЗДІЛУ 33

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ 34

5.1 КЕРІВНИЦТВО КОРИСТУВАЧА 34

5.2 ВИПРОБУВАННЯ ПРОГРАМНОГО ПРОДУКТУ 44

5.2.1 Мета випробувань 44

5.2.2 Загальні положення 45

5.2.3 Результати випробувань 45

ВИСНОВОК ДО РОЗДІЛУ 48

ЗАГАЛЬНІ ВИСНОВКИ 49

ПЕРЕЛІК ПОСИЛАНЬ 50

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ 51

ВСТУП

Безліч компаній та підприємств мають більше ніж одну систему, де вони зберігають та оброблюють дані, найчастіше це стається через те, що різне програмне забезпечення допомагає у вирішенні різного типу задач для різних підрозділів. Але ці данні часто потрібно синхронізувати, переміщувати між системами та трансформувати під кожну конкретну систему.

Корпоративна шина даних або Сервісна шина підприємства - сполучне програмне забезпечення, що забезпечує централізований та уніфікований, орієнтований на події обмін повідомленнями між різними інформаційними системами на принципах сервіс-орієнтованої архітектури. Поняття введено на початку 2000-х років фахівцями підрозділу Progress Software - Sonic, які розробляли програмне забезпечення, орієнтоване на обробку повідомлень – SonicMQ [1].

Інтеграція – процес об'єднання різних обчислювальних систем і програмних застосунків фізично або функціонально [2]. Системна інтеграція полягає у розробці комплексних рішень, призначених для досягнення максимальної ефективності функціонування системи шляхом налагодження ефективної взаємодії її підсистем, обміну даними.

Дипломний проєкт присвячений розробці сервісу інтеграцій програмного забезпечення.

Практичне значення одержаних результатів. Розроблено програмне забезпечення та алгоритм перенесення даних динамічно-заданих типів.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Інтеграція програмного забезпечення складається з таких етапів:

1. Імпорт даних з системи А.
2. Трансформування даних з типу системи А у тип системи Б.
3. Експорт даних у систему Б.

Різні системи можуть мати різну структуру даних, формат даних (JSON, XML, і т.п.), як і точки взаємодії з ними (API, FTP-сервер, і т.п.). Перетворення даних та експорт, імпорт їх із системи в систему є основними завданнями будь-якої інтеграції.

Типи формату даних та типи точок взаємодії повторюються і їх можна задати програмно з можливістю конфігурування, а структура даних у кожної системи індивідуальна і повинна бути налаштована користувачем або динамічно (виходячи з готового об’єкту). Перетворення динамічних (або налаштованих) даних у інші динамічні дані є одним з найскладніших завдань, зазвичай їх пишуть вручну для кожного типу даних. Оскільки об’єкти можуть мати вкладення та масиви, повинна бути певна логіка обробки та виставлення співвідношень у таких типів.

Також важливою частиною інтеграції є зберігання результатів, тобто – транзакцій з усіма необхідними даними щоб знайти і ідентифікувати об’єкт у обох системах.

1.1.1 Опис процесу діяльності

Процес інтеграції залежить від систем, що приймають участь в інтеграції та типів. Щоб сервіс підтримував певну систему, необхідно мати про неї та тип усі необхідні дані, також дані про зв’язки з типом експорту даних.

Оскільки форматів даних та типів точок доступу існує декілька, сервіс надає можливість обрати з наявних підтримуваних форматів.

На Рисунку 1.1 зображено діаграму діяльності сайту

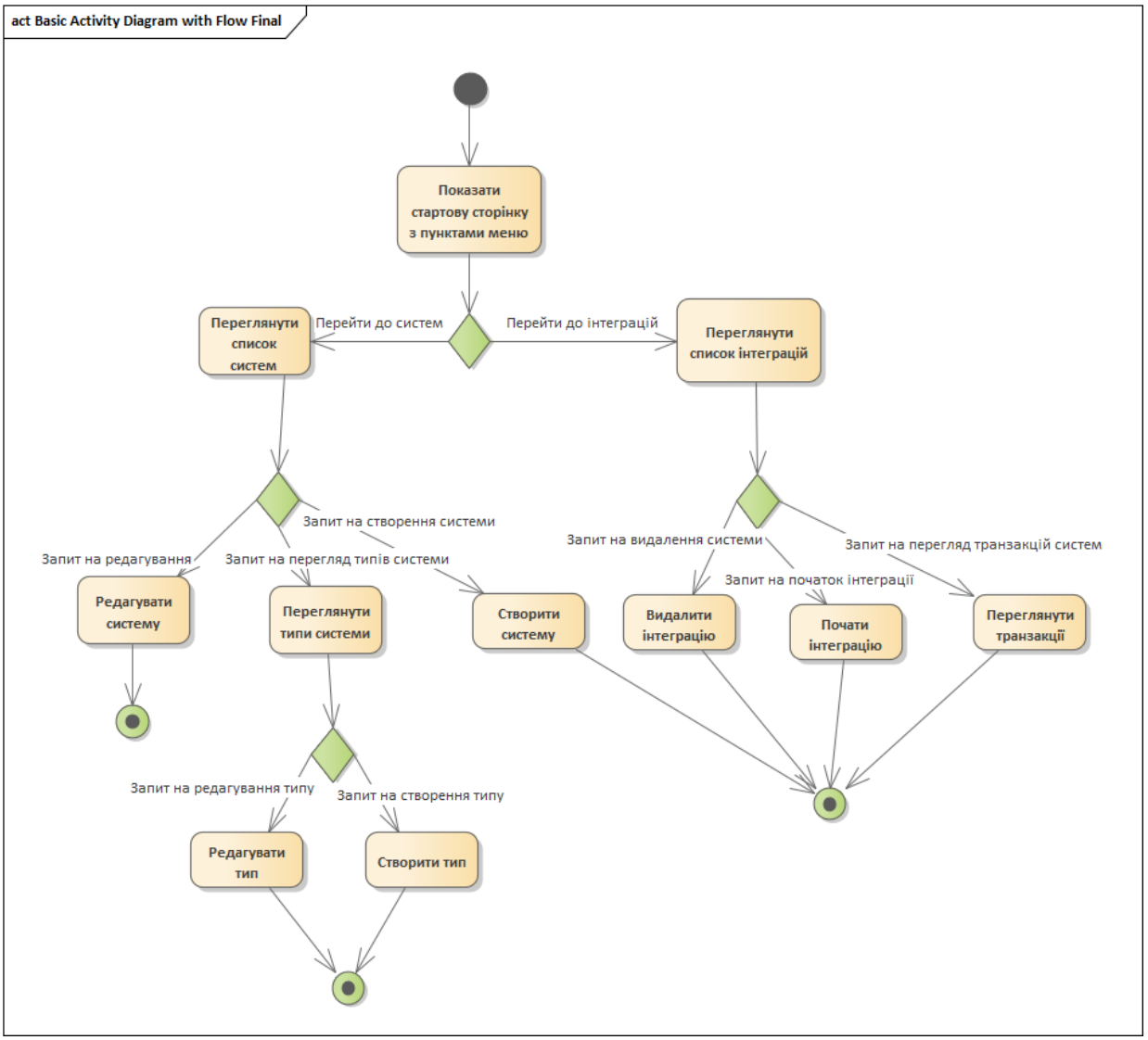


Рисунок 1.1 – Діаграма діяльності сервісу

1.1.2 Опис функціональної моделі

Актором сервісу є технічний спеціаліст. Дії або варіанти використання, що виконує в сервісі актор, наведені в таблиці 1.1, в якій описаний актор, варіанти використання та їх описи дій.

Таблиця 1.1 – Типи залежностей між варіантами використання.

Актор	*Варіант використання*	*Опис дії варіанта використання*
Технічний спеціаліст	Налаштовування систем інтеграцій	Технічний спеціаліст створює, редагує данні, точок доступу, формату даних систем, з якими проводяться інтеграції.
	Налаштовування типів інтеграцій	Технічний спеціаліст створює, редагує данні типів та полів, точок доступу до типів систем.
	Налаштовування інтеграцій	Технічний спеціаліст створює інтеграції систем, їх зв’язки та вводить данні, налаштовані у системах для встановлення зв’язку.
	Керування транзакціями	Технічний спеціаліст контролює транзакції, утворені під час інтеграції.

Відповідно визначених варіантів використання представлено загальну модель варіантів використання на Рисунку 1.2.

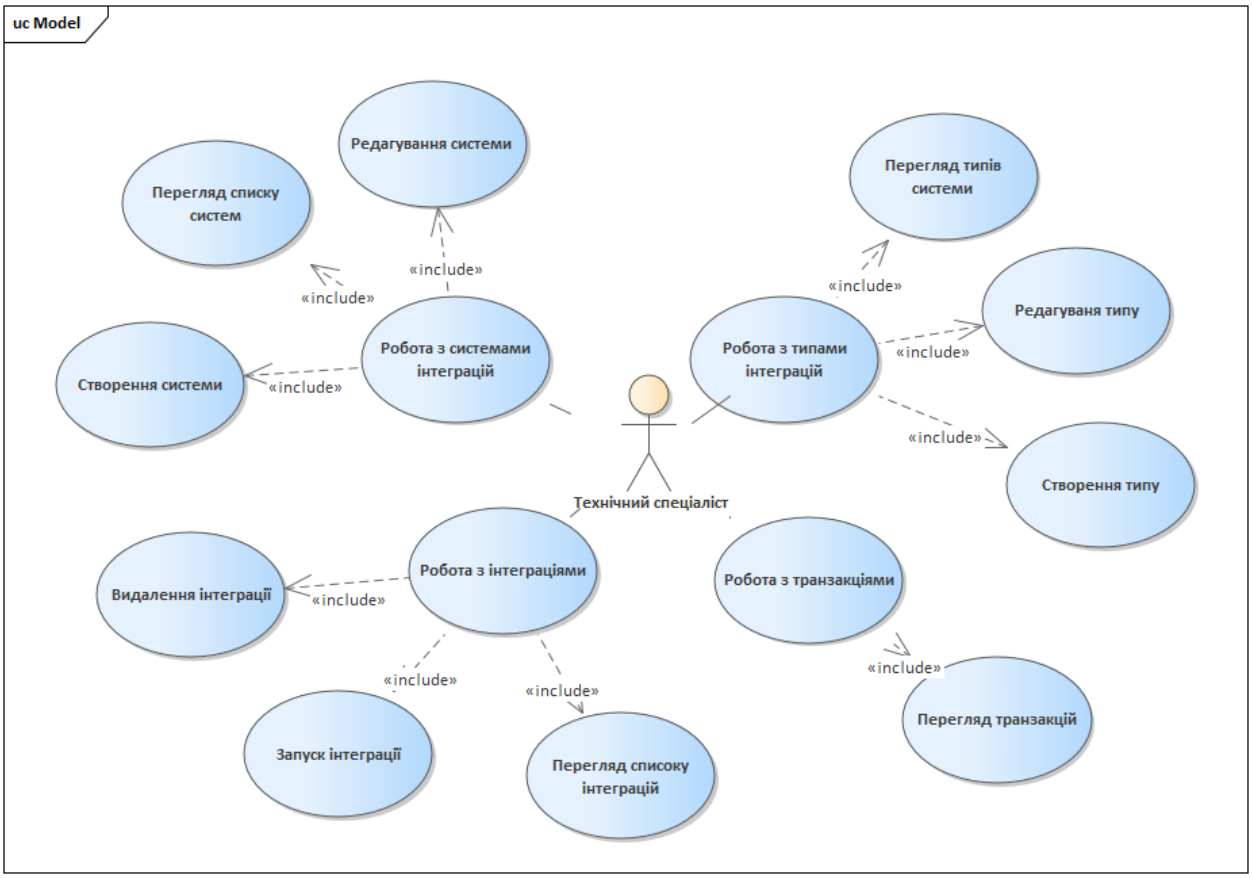


Рисунок 1.2 – Схема структурна варіантів використання

1.2 Огляд наявних аналогів

Проаналізувавши різні системи, що мають точки взаємодії для інтеграцій, можемо сказати що інтеграційне програмне забезпечення може бути, як настільним додатком на комп'ютері так і веб сайтом.

Bezala – система, у вигляді веб-сторінки, що дозволяє інтегруватися з обмеженою кількістю банківських систем. Тобто, для кожної системи та трансформації спеціально пишеться код розробниками.

Oracle Service Bus – система, що має вигляд додатку для різних операційних систем та має великий функціонал з інтегрування різноманітних сервісів. Але по-перше ця система має дуже високий поріг входу знань для користувача, по-друге змінює архітектуру, що робить цю систему єдиним

джерелом обробки інформації з різних систем, погано працює із вкладеними типами і масивами (наприклад, якщо у типа є вкладені типи, та деякі поля є масивами, що потребують фільтрації).

Інші системи подібні до Bezala або Oracle Service Bus, тож розбір цих двох систем є показовим.

Отже, кожен з аналогів не має таких функціональних можливостей, як:

- робота з динамічно-заданими типами, у тому числі із вкладеними типами і масивами, фільтрацією внутрішніх масивів, трансформація таких даних;
- робота з сервісом без зміни архітектури програмних забезпечень підприємства і іншої комунікації між ними;
- зрозумілий і легкий у використанні інтерфейс.

Тому необхідна розробка свого сервісу інтеграції програмного забезпечення.

1.3 Постановка задачі

1.3.1 Призначення розробки

У зв’язку з тим, що більшість підприємств мають багато систем обробки даних і намагаються синхронізувати їх через написання власних систем для інтеграції конкретних, з’явилась мета зробити гнучкий та динамічний сервіс інтеграцій, що не буде потребувати написання коду на підприємствах під конкретні системи, а саме: створити застосунок, який буде виступати в ролі точки імпорту, експорту та трансформації даних для різних систем. Технічний спеціаліст, що буде налаштовувати сервіс, зможе вказати всі точки доступу (системи), типи та створити інтеграції. Всі результати інтеграцій будуть зберігатись як транзакції, щоб технічний спеціаліст бачив, які данні були перенесені сервісом.

1.3.2 Цілі та задачі розробки

Метою розробки є збільшення можливостей інтеграції програмного забезпечення завдяки створенню динамічних типів та систем, конфігурування точок доступу, налаштовуванню відношень між типами програмного забезпечення.

Призначення розробки є:

- динамічне налаштування точок доступу до систем;

- динамічне налаштування типів даних систем та встановлення зв’язків між даними різних систем;

- імпорт, експорт даних систем та трансформування даних.

Для досягнення поставлених цілей необхідно реалізувати наступні задачі:

- розробити комфортний та зрозумілий інтерфейс для технічного спеціаліста;

- спроєктувати базу даних для збереження налаштувань систем, типів, інтеграцій та транзакцій.

Реалізація цих задач дозволить створити зручне середовище для налаштування інтеграцій програмного забезпечення.

Висновок до розділу

Здійснено детальний аналіз предметної області і визначено актора системи, - технічного спеціаліста. Також у розділі було оглянуто та проаналізовано існуючі аналоги, продемонстровано недоліки існуючих програмних продуктів.

Визначено мету, цілі та задачі розробки. Було описано процес діяльності розробленої інформаційної системи.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Таблиця 2.1 «Вхідні дані».

Дані	*Опис*
Дані системи	Основні дані, необхідні для інтеграції з системою (назва, url, формат даних, варіант доступу до системи, тощо)
Тип системи	Дані типу, для налаштування трансформування даних.
Поля типу системи	Дані поля типу, для налаштування трансформування даних (назва, тип поля, чи є масивом, тощо).
Дані зв’язку з системою	Дані полів, їх роль у запиті для встановлення доступу до системи.
Дані зв’язку типів	Дані, що визначають зв’язок між полями типів для налаштування інтеграцій.
Дані інтеграції	Дані типів та значення полів зв’язку з системами для проведення інтеграції.

2.2 Вихідні дані

Таблиця 2.2 «Вихідні дані».

Дані	*Опис*
Дані транзакції	Дані результатів інтеграції.
Утворені файли або об’єкти	Об’єкти або файли, експортовані системою.

2.3 Опис структури бази даних

Для зберігання даних у проєкті використовується реляційна база даних з такими таблицями: Integration, Transaction, TypeDefinition, ConnectionFieldValue, ConnectionFieldDefinition, FieldDefinition, FieldConnection.

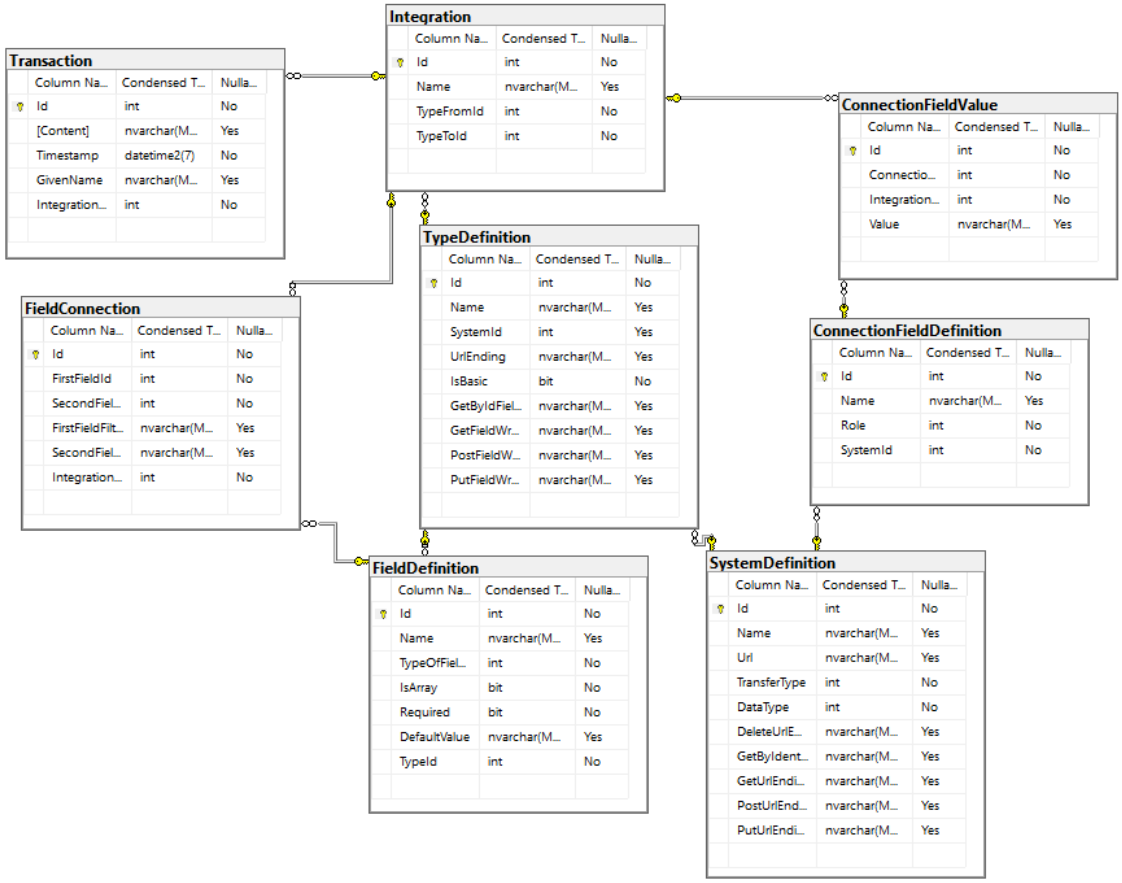


Рисунок 2.1 – Фізична модель бази даних

Таблиця 2.3 «Опис таблиць бази даних».

Назва таблиці	Назва параметру	Тип даних	Опис
Transaction	**Id**	INT	Первинний ключ, унікальне значення

Продовження таблиці 2.3

Назва таблиці	Назва параметру	Тип даних	Опис
Transaction	**Content**	NVARCHAR	Кінцевий результат трансформації об’єкту
	Timestamp	DATETIME	Дата та час створення транзакції
	GivenName	NVARCHAR	Ім’я файлу/транзакції, згенероване системою
	IntegrationId	INT	Зовнішній ключ інтеграції, не пусте значення
Integration	**Id**	INT	Первинний ключ, унікальне значення
	Name	NVARCHAR	Назва інтеграції для зручності користувача
	TypeFromId	INT	Зовнішній ключ типу для імпорту, не пусте значення

Продовження таблиці 2.3

Назва таблиці	Назва параметру	Тип даних	Опис
Integration	**TypeToId**	INT	Зовнішній ключ типу для експорту, не пусте значення
ConnectionFieldValue	**Id**	INT	Первинний ключ, унікальне значення
	Value	NVARCHAR	Значення поля встановлення зв’язку з системою
	ConnectionFieldId	INT	Зовнішній ключ поля зв’язку з системою, не пусте значення
	IntegrationId	INT	Зовнішній ключ інтеграції, не пусте значення
ConnectionFieldDefinition	**Id**	INT	Первинний ключ, унікальне значення
	Name	NVARCHAR	Назва поля встановлення зв’язку з системою

Продовження таблиці 2.3

Назва таблиці	Назва параметру	Тип даних	Опис
ConnectionFieldDefinition	**Role**	INT	Роль поля у запиті до системи, не пусте значення
	SystemId	INT	Зовнішній ключ системи, не пусте значення
SystemDefinition	**Id**	INT	Первинний ключ, унікальне значення
	Name	NVARCHAR	Назва системи для зручності користувача
	Url	NVARCHAR	Адреса доступу до системи (http/ftp)
	TransferType	INT	Формат зв’язку з системою (api/ftp), не пусте значення
	DataType	INT	Формат даних системи (json/xml), не пусте значення

Продовження таблиці 2.3

Назва таблиці	Назва параметру	Тип даних	Опис
SystemDefinition	**DeleteUrlEnding**	NVARCHAR	Закінчення адреси доступу для видалення (папка на ftp, метод контролера, тощо)
	GetUrlEnding	NVARCHAR	Закінчення адреси доступу для отримання всіх об’єктів (папка на ftp, метод контролера, тощо)
	GetByIdUrlEnding	NVARCHAR	Закінчення адреси доступу для отримання конкретного об’єкту (папка на ftp, метод контролера, тощо)

Продовження таблиці 2.3

Назва таблиці	Назва параметру	Тип даних	Опис
SystemDefinition	**PostUrlEnding**	NVARCHAR	Закінчення адреси доступу для створення конкретного об’єкту (папка на ftp, метод контролера, тощо)
	PutUrlEnding	NVARCHAR	закінчення адреси доступу для зміни конкретного об’єкту (папка на ftp, метод контролера, тощо)
TypeDefinition	**Id**	INT	Первинний ключ, унікальне значення
	Name	NVARCHAR	Назва типу для зручності користувача
	SystemId	INT	Зовнішній ключ системи

Продовження таблиці 2.3

Назва таблиці	Назва параметру	Тип даних	Опис
TypeDefinition	**UrlEnding**	NVARCHAR	Закінчення адреси доступу до типу (папка на ftp, метод контролера, тощо)
	IsBasic	BIT	Ідентифікує чи є тип базовим, не пусте значення
	GetFieldWrapper	NVARCHAR	Поле, що обгортає тип на відповіді (наприклад, data або cathegories)
	GetByIdFieldWrapper	NVARCHAR	Поле, що обгортає тип на відповіді (наприклад, data або cathegory)
	PostFieldWrapper	NVARCHAR	Поле, що обгортає тип на відповіді (наприклад, data або cathegory)

Продовження таблиці 2.3

Назва таблиці	Назва параметру	Тип даних	Опис
	PutFieldWrapper	NVARCHAR	Поле, що обгортає тип на відповіді (наприклад, data або cathegory)
FieldDefinition	**Id**	INT	Первинний ключ, унікальне значення
	Name	NVARCHAR	Назва поля в об’єкті
	TypeOfFieldId	INT	Зовнішній ключ типа поля, не пусте значення
	IsArray	BIT	Ідентифікує чи є поле масивом, не пусте значення
	Required	BIT	Ідентифікує чи є поле обов’язковим для заповнення, не пусте значення
	DefaultValue	NVARCHAR	Стандартне значення поля

Продовження таблиці 2.3

Назва таблиці	Назва параметру	Тип даних	Опис
	TypeId	INT	Зовнішній ключ типа, якому належить поле, не пусте значення
FieldConnection	**Id**	INT	Первинний ключ, унікальне значення
	FirstFieldFilterFunction	NVARCHAR	Функція фільтрації, -перше поле масив
	SecondFieldFilterFunction	NVARCHAR	Функція фільтрації, якщо друге поле масив
	FirstFieldId	INT	Зовнішній ключ першого поля, не пусте значення
	SecondFieldId	INT	Зовнішній ключ другого поля, не пусте значення
	IntegrationId	INT	Зовнішній ключ інтеграції, не пусте значення

Висновок до розділу

У розділі були описані вихідні та вхідні дані, структура бази даних, було наведено пояснення до кожної таблиці з типами даних та описом до них.

В рамках проєктування бази даних було спроєктовано 8 таблиць та 12 зовнішніх ключів.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Найскладнішим етапом інтеграції є трансформування даних, зазвичай, коли програмісти інтегрують одну систему з іншою – вручну пишуть яке поле с типу А перенести у поле типу Б.

Оскільки наша система розрахована на конфігурацію типів, які з'єднуються між собою лише базовими типами, то ми під час трансформування даних виконуємо пошук шляху від поля, що з'єднано з шуканим до самого типу. Базовими типами є текстовий рядок, число та бінарний тип (string, number, boolean).

Приклад варіанту зв'язку між типами при проведенні інтеграції з типу Б у тип А зображено на Рисунку 3.1.

Рисунок 3.1 – Приклад зв'язку між типами

3.2 Математична постановка задачі

Припустимо, що кожен тип (у тому числі базовий) є вершиною графа. Зобразимо тип А і тип Б з Рисунку 3.1 у вигляді окремих графів.

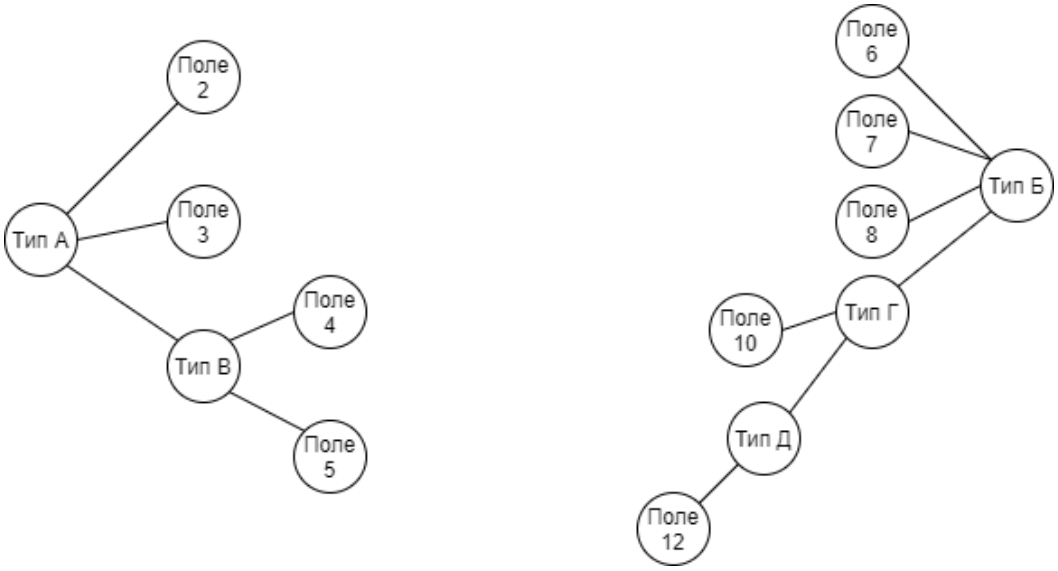


Рисунок 3.2 – Типи у вигляді графів

Таким чином ми бачимо, що типи для інтеграції є деревами, у яких базові типи – листки, тому що:

Дерево - скінченна множина T одного або більше вузлів з наступними властивостями:

1. Існує один корінь дерева T (в нашому випадку Тип А та тип Б є коренями дерев).
2. Інші вузли (за винятком кореня) розподілені серед M ≥ 0 непересічних множин $T_1,...,T_m$ і кожна з множин є деревом; дерева $T_1,...,T_m$ називаються піддеревами даного кореня T.

У нашому випадку множини M – підтипи (Тип В, Тип Г, Тип Д).

Але перед інтеграцією ми встановлюємо зв’язки між полями базових типів, тож дерева становляться підграфами основного графу трансформації даних.

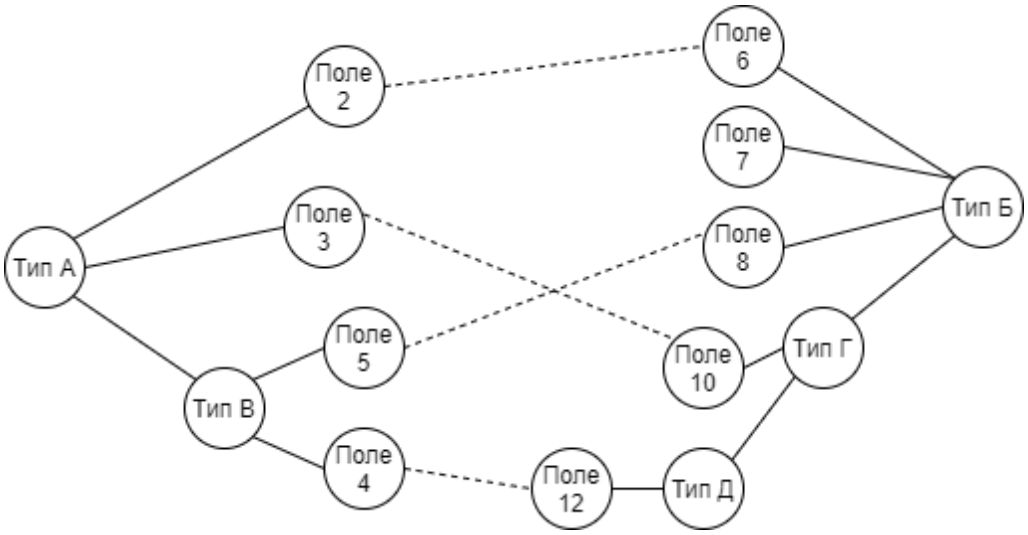


Рисунок 3.3 – Типи у вигляді графів зі зв’язками між полями

Для проведення трансформації нам потрібна буде лише зв’язна частина графа, тобто вершина «Поле 7» нам не знадобиться.

Наша задача – під час проходження листків дерева підграфу Типу А при наявності ребер, що не належать дереву знайти шлях до кореня дерева Б. Тобто потрібно знайти шлях або простий ланцюг від листків дерева Типу А до кореня дерева Типу Б. Оскільки у нас немає додаткових умов, можемо вважати що вага кожного ребра = 1. У вигляді математичних означень це буде виглядати так:

$G = (V, E)$ – граф трансформації даних

$S_1 \subset V$ — підмножина вершин графа G, що відносяться до типу А

$S_2 \subset V$ — підмножина вершин графа G, що відносяться до типу Б

Підграфи $G[S_1]$, $G[S_2]$ — це графи, вершинами яких є елементи S_1, S_2 а ребра яких складаються з усіх ребер з множини E кінцеві вершини яких належать S_1, S_2

Вагою w(p) шляху $p = \langle v_0, v_1, \ldots, v_k \rangle$ назвемо суму ваг ребер, що входять в цей шлях:

$$\omega(p) = \sum_{i=1}^{k} \omega(v_{i-1}, v_i) \qquad (3.1)$$

Задача також ускладнюється, якщо хоча б одне поле, що входить у зв’язний підграф є масивом.

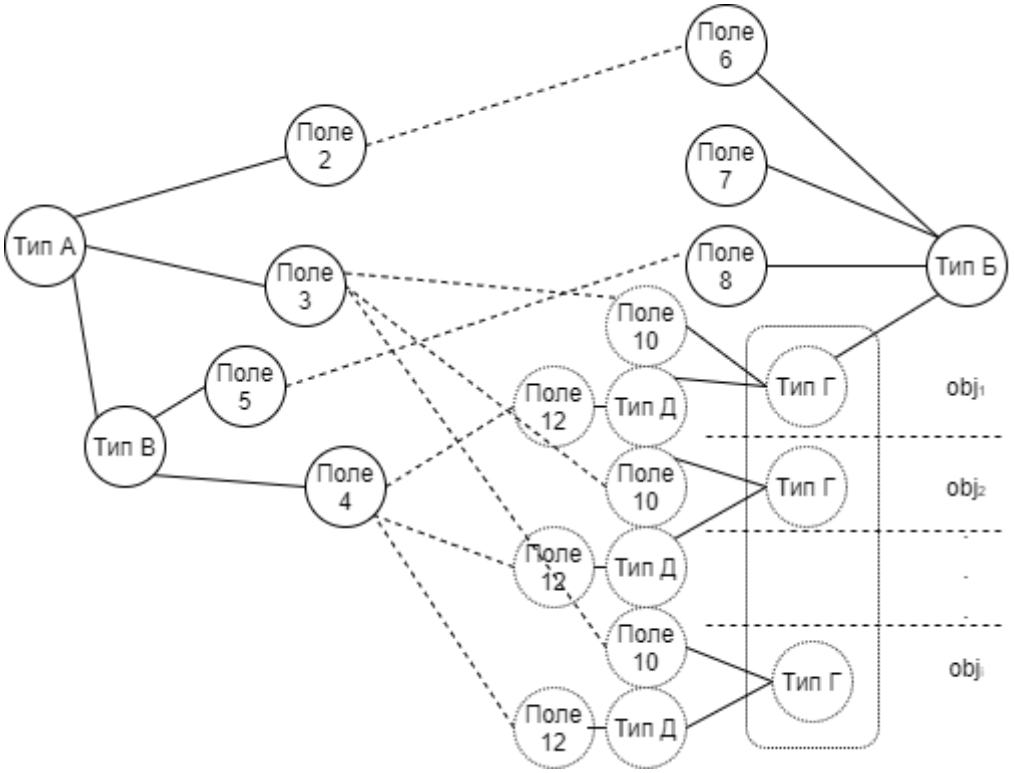


Рисунок 3.4 – Вигляд графу типів при наявності масиву (Поле 9 Типу Г)

Тоді ребрам між деревами потрібно надати різну вагу. За алгоритмом вагу ребру може задати користувач використовуючи функції фільтрування масиву (тобто ми видаляємо інші вершини або робимо вагу їх ребрам більшу, ніж 1). Якщо функцію не задано користувачем, то ми беремо перший елемент.

3.3 Обґрунтування методу розв’язання

Для пошуку найкоротших шляхів у зваженому графі існують такі алгоритми, як: A*, Дейкстри, Беллмана-Форда, Флойда-Воршелла.

Оскільки у нас немає ребер з негативною вагою і ми шукаємо алгоритм з найменшою складністю O, то було обрано алгоритм A* зі складністю: O(|V|*log |V|)

3.4 Опис методів розв’язання

Метод розв’язання полягає у тому, щоб знайти від листка дерева типа, у який трансформується об’єкт, що є підграфом основного графа найкоротший шлях до кореня дерева типа, з якого відбувається трансформація, що також є підграфом основного графа.

В алгоритмі A* розпізнаються вершини, яким співставляються значення f(x).

Функція f(x) для вершини x визначається так:

$f(x)=g(x) + h(x)$ (3.2)

де:

g(x) – вартісна функція шляху від початкової вершини до x;

h(x) евристична функція, що надає оцінку шляху від вершини x до кінцевої.

Для кожного з сусідів оновлюється відстань, значення евристичної функції і він додається в безліч Q [6].

Висновок до розділу

У розділі математичного забезпечення описані математичні задачі дипломного проєкту, розглянуте представлення типів як дерев, що з’єднуються у граф трансформації даних за допомогою відношень між типами.

Оглянуті різні варіанти вирішення поставлених задач. Обґрунтовано вибір поточних рішень.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Програмне забезпечення, що використовується при розробці:

- ОС Windows [7];
- веб-браузер Google Chrome [8].
- IDE Visual Studio 2019.
- мови написання коду програми : C# - об’єктно-орієнтована мова програмування [10].
- каркаси застосунків: ASP.NET Core — вільне та відкрите програмне забезпечення каркаса веб-застосунків, розроблена корпорацією Microsoft і співтовариством, має модульну структуру, яка працює як на повній платформі .NET Framework, так і на платформі .NET Core [3], NestJS - основа для створення ефективних масштабованих веб-додатків Node.js, розроблена компанією Trilon, Angular [4].

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Вимоги до серверу:

– ОЗУ об’ємом 2,5 ГБ;

– центральний процесор з частотою не менше 2,0 ГГц;

– графічна карта повинна підтримувати DirectX.

– вільного простору на накопичувачі на магнітних дисках 20 Гб і частотою обертання не менше 5 400 об/хв;

Також необхідне встановлене наступне програмне забезпечення:

- Node.js 12 + npm 6.14;
- глобально npm пакети: @nestjs/cli 6.14.0, @angular/cli 9.1.5;
- .NET Core 3, ASP.NET Core 3 Runtime + Hosting;

- SQLServer;
- Windows 10 або Ubuntu 18.04.

Вимоги до клієнта:

- Safari 12.0+;
- Firefox 66.0+;
- Google Chrome 74.0+;
- Chromium 71+.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

Схема структурна класів програмного забезпечення представлена у графічних матеріалах.

4.3.2 Діаграма послідовності

Діаграма послідовності є різновидом діаграми в UML. Зазвичай, одна діаграма послідовності присвячена опису одного з варіантів використання, зазначеного у Use-case діаграмі.

У графічному матеріалі представлена схема структурна послідовності проведення інтеграції. Технічний спеціаліст обирає інтеграцію для запуску та починає інтеграцію. Портал надсилає повідомлення до сервера для запуску певної інтеграції, сервер знаходить всі дані щодо інтеграції у базі даних та починає проходження етапів інтеграції обмінюючись повідомленнями із сервісом інтеграції. Після завершення всіх етапів транзакції зберігаються у базі даних і відображаються технічному спеціалісту.

4.3.3 Діаграма компонентів

Схема структурна компонентів представлена у графічних матеріалах.

4.3.4 Специфікація функцій

Функції системи описані у таблиці 4.1.

Таблиця 4.1 «Специфікація функцій»

Назва функції	*Дія*
RunIntegration(int id)	Почати інтеграцію за заданим id.
fetch (fetchType: FetchType, type?: TranspherType, data: any)	Забрати дані з джерела
parse(data: any, type?: DataType)	Поточний формат даних змінити на JSON
map(objFrom: any, connections: FieldConnection[], typeFromId: number, typeToId: number, types: TypeDefinition[])	Трансформування об’єкту за заданими типами та зв’язками.
write(data: any, type?: DataType)	Формат даних JSON змінити на формат даних системи.
generateTypes(data: any, name?: string, addDefaultValues: boolean)	Генерація типів за заданим об’єктом.
generateConnections(objFrom: any, objTo: any, typeFromId: number, typeToId: number, types: TypeDefinition[])	Генерація зв’язків за заданими об’єктами та типами.

Висновок до розділу

В розділі програмного та технічного забезпечення були розглянуті засоби та фреймворки для реалізації програмного продукту. Наведено опис

діаграм із детальним поясненням до них, що наведені у графічних матеріалах. Наведені технічні вимоги для роботи програмного забезпечення.

Наведено специфікацію основних функцій системи з поясненням до них.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

При включенні програмного продукту відкривається головна сторінка з основними підказками щодо роботи з системою, представлена на Рисунку 5.1.

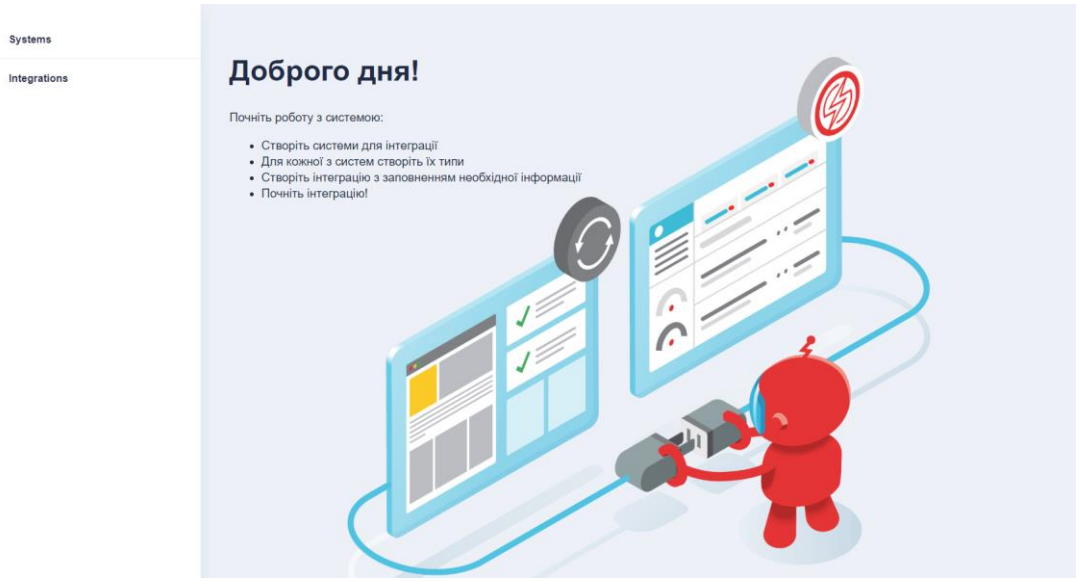


Рисунок 5.1 – Головна сторінка

Для того, щоб створити нову систему для інтеграцій необхідно натиснути на пункт меню «Systems», відкриється сторінка з усіма створеними системами, що представлена на Рисунку 5.2.

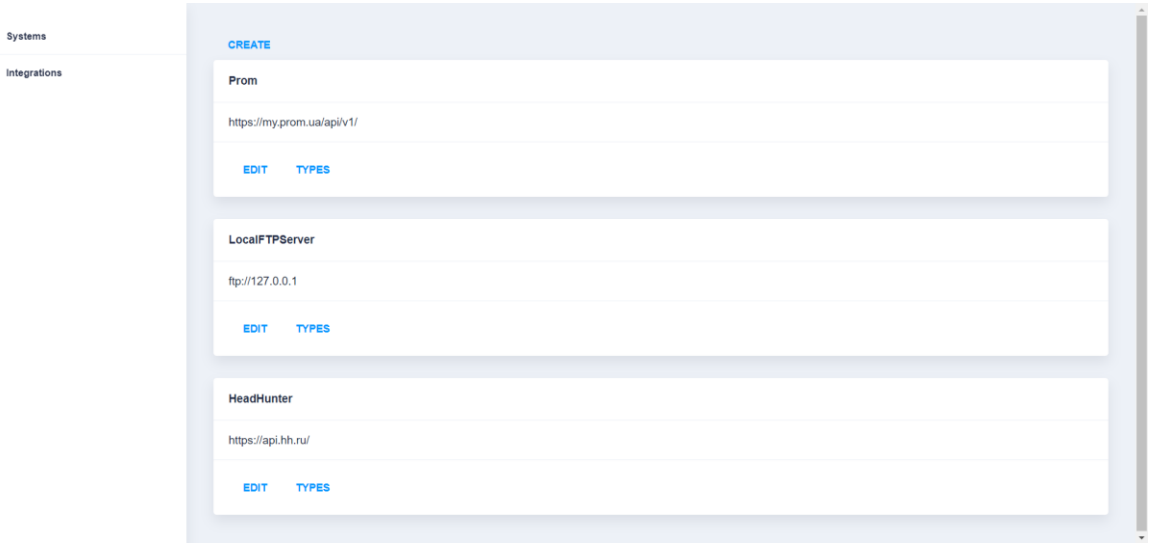


Рисунок 5.2 – Сторінка перегляду систем

На сторінці систем потрібно натиснути кнопку «Create». Тоді відкриється сторінка створення системи, що продемонстрована на Рисунку 5.3.

Рисунок 5.3 – Сторінка створення системи

На даній сторінці необхідно заповнити усі поля, що відповідають за з'єднання з певною системою:

Name – назва системи для зручності користувача;

Source Type - формат зв’язку з системою (api/ftp);

Data Type - формат даних системи (json/xml);

Url - адреса доступу до системи (http/ftp);

Get Url Ending - закінчення адреси доступу для отримання всіх об’єктів;

Post Url Ending - закінчення адреси доступу для створення конкретного об’єкту;

Delete Url Ending - закінчення адреси доступу для видалення (папка на ftp, метод контролера, тощо);

Get By Identifier Url Ending - закінчення адреси доступу для отримання конкретного об’єкту;

Put Url Ending - закінчення адреси доступу для зміни конкретного об’єкту.

Кнопка «Add connection field» відповідає за створення поля доступу до системи. Після натиснення цієї кнопки з’являються нові поля для вводу, що продемонстровані на Рисунку 5.4.

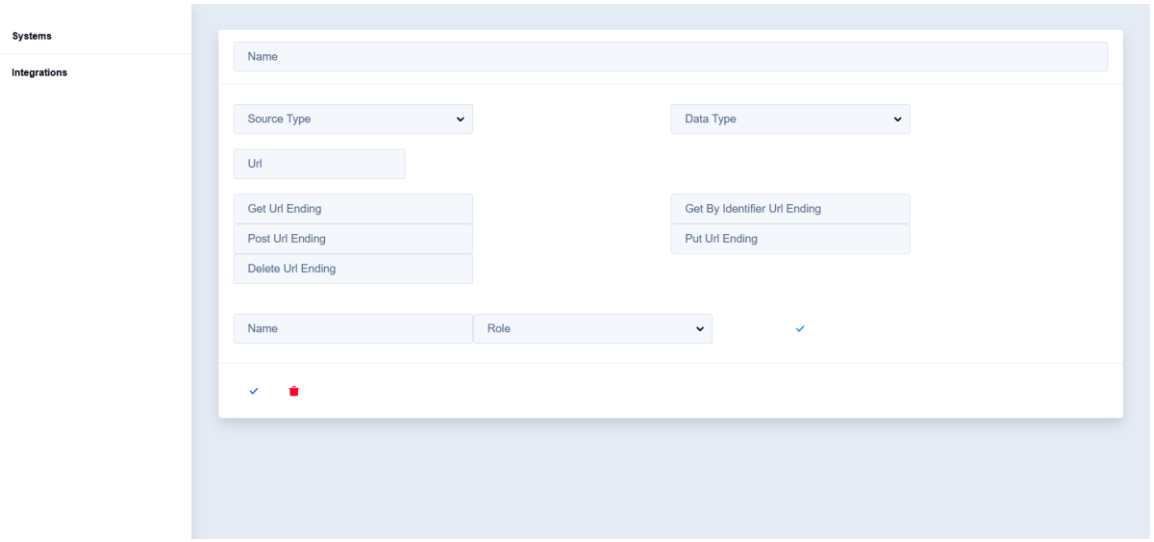


Рисунок 5.4 – Сторінка створення системи з полями створення полів доступу

Name – назва поля;

Role – роль поля у запиті (headers, body, query).

Приклад заповнених полів системи показано на Рисунку 5.5.

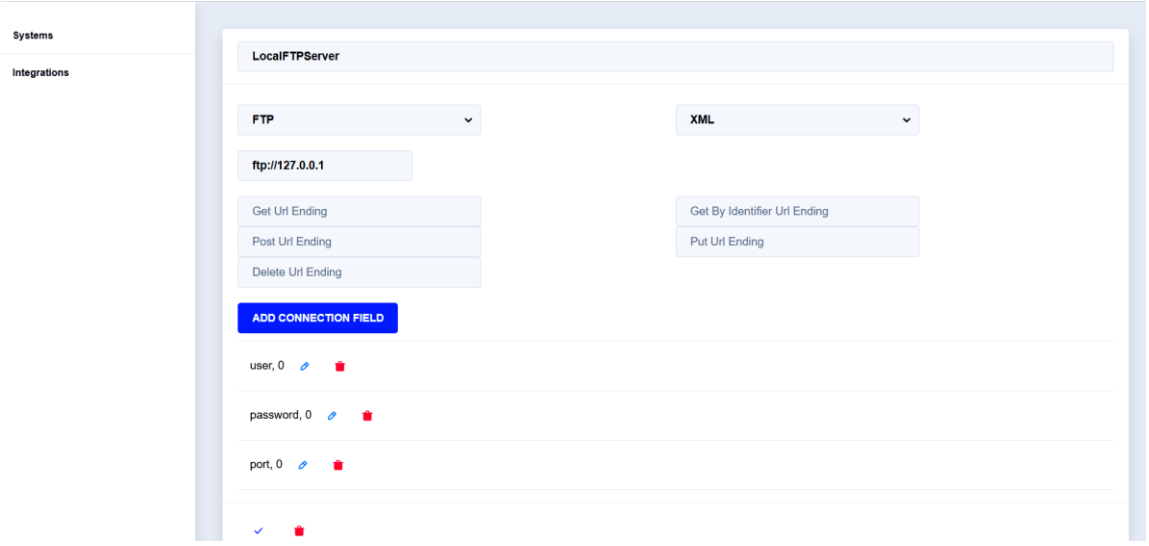


Рисунок 5.5 – Приклад заповнення полів системи

Для закінчення створення або редагування інтеграції необхідно натиснути галочку. Щоб видалити тип – корзину. Після створення системи

необхідно створити тип системи, з яким буде проходити імпорт, експорт та трансформація даних. Створені системи відображаються на сторінці «Systems», що показана на Рисунку 5.2. Для перегляду типів системи потрібно натиснути на ній на кнопку «Types». Відкриється сторінка з типами системи, що показана Рисунку 5.6.

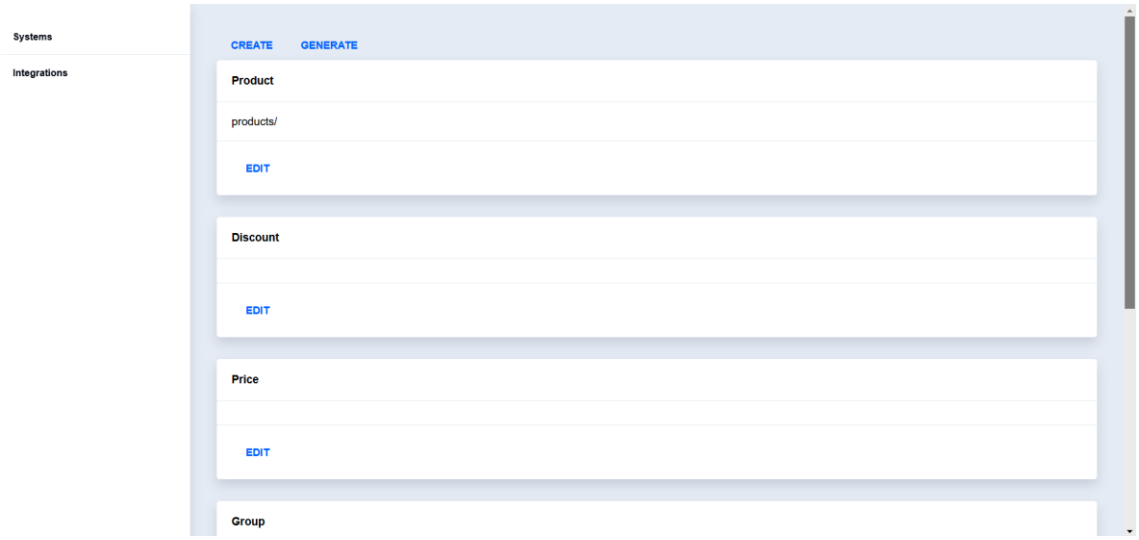


Рисунок 5.6 – Перегляд типів системи

Щоб створити новий тип необхідно натиснути на кнопку «Create», відобразиться сторінка створення типу, що продемонстрована на рисунку 5.7.

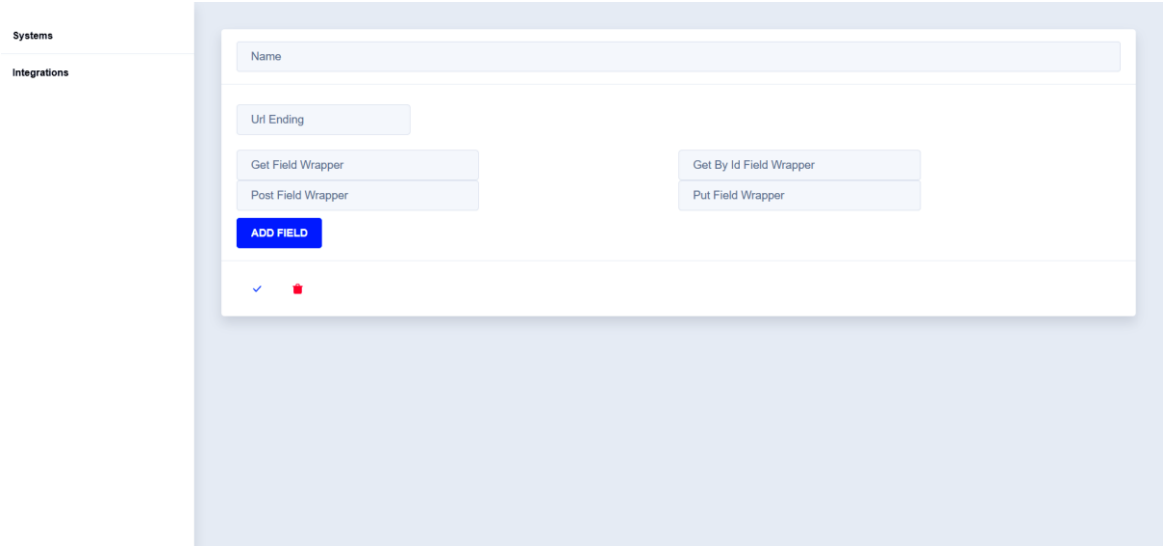


Рисунок 5.7 – Створення типу системи

На даній сторінці необхідно заповнити усі поля, що відповідають за створення схеми типу.

Name - назва типу для зручності користувача;

UrlEnding - закінчення адреси доступу до типу (папка на ftp, метод контролера, тощо) ;

GetFieldWrapper - поле, що обгортає тип на відповіді запиту всіх об'єктів;

GetByIdFieldWrapper - поле, що обгортає тип на відповіді запиту за ідентифікатором;

PostFieldWrapper - поле, що обгортає тип на відповіді створення об'єкту;

PutFieldWrapper - поле, що обгортає тип на відповіді зміни об'єкту.

Кнопка «Add field» відповідає за створення поля у типі. Після натиснення цієї кнопки з’являються нові поля для вводу, що продемонстровані на Рисунку 5.8.

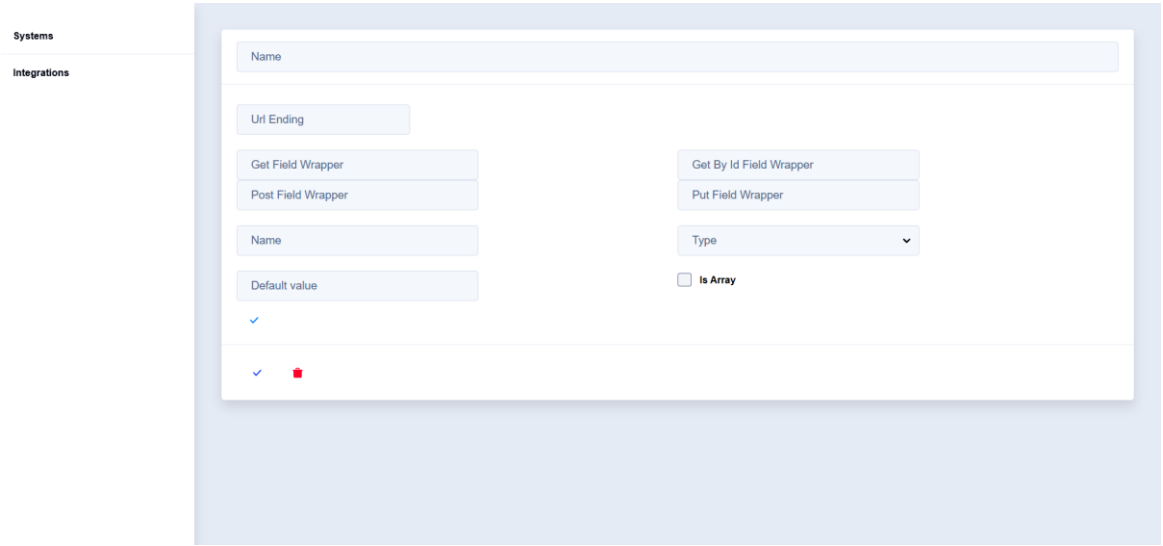


Рисунок 5.9 – Створення типу системи і його полів

Name – назва поля;

Type – тип поля (базовий чи один з типів системи);

Default value – значення поля по замовчуванню;

Is Array – ідентифікує чи є поле масивом.

Приклад заповнених полів системи показано на Рисунку 5.10.

Рисунок 5.10 – Приклад заповнених полів типу

Для закінчення створення або редагування системи необхідно натиснути галочку. Щоб видалити систему – корзину. Після створення систем і типів потрібно створити інтеграцію. Щоб переглянути наявні інтеграції необхідно перейти на вкладку «Integrations».

Рисунок 5.11 – Сторінка перегляду інтеграцій

Для створення нової інтеграції необхідно натиснути на кнопку «Create».

Рисунок 5.12 – Створення нової інтеграції

Спочатку відображаються лише поля вибору систем для інтеграції та назва.

Name – назва інтеграції для зручності користувача;

System from – система імпорту даних;

System to – система експорту даних.

Після вибору систем з’являються поля вибору типів з цих систем та поля з’єднання з системами.

Рисунок 5.13 – Сторінка створення нової інтеграції після обрання систем

Type from – тип імпорту;

Type to – тип експорту.

Тепер необхідно обрати типи імпорту та експорту.

Рисунок 5.13 – Сторінка створення нової інтеграції після обрання типів

Після обрання типів їх поля будуть завантажені на екран. Для створення зв'язків між типами. Щоб створити тип потрібно натиснути на кнопку «Add connection» та обрати поля з обох типів.

Рисунок 5.14 – Створення зв'язку між типами

First Field Filter Function – функція фільтрації, якщо перше поле масив;

Second Field Filter Function - функція фільтрації, якщо друге поле масив.

Для завершення створення зв’язку необхідно натиснути кнопку «Save».

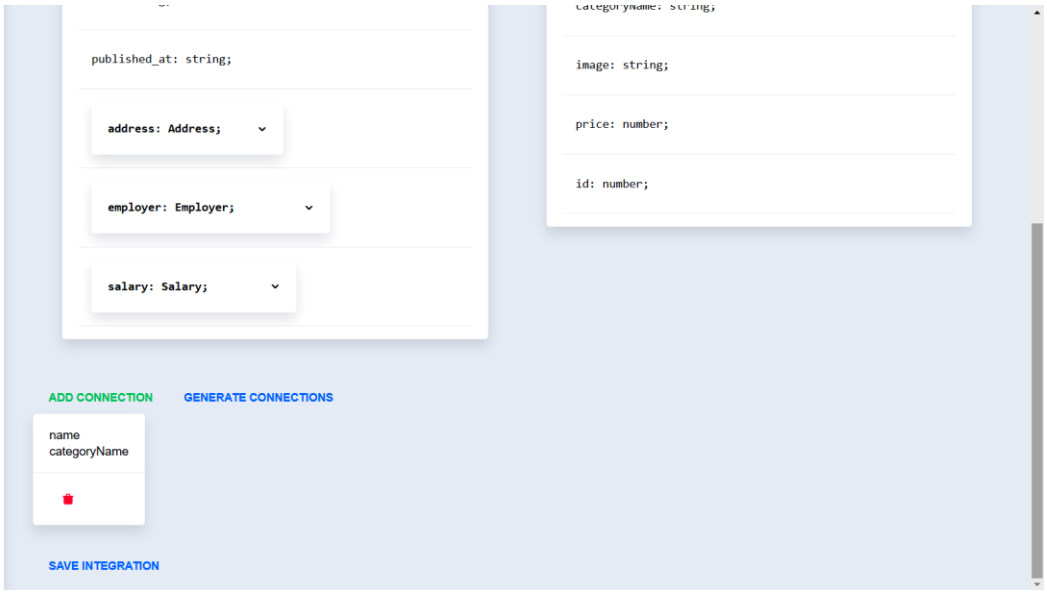


Рисунок 5.15 – Вигляд створених зв’язків

Зв’язки можна видаляти натиснувши на корзину. Також зв’язки можна згенерувати натиснувши на кнопку «Generate connections».

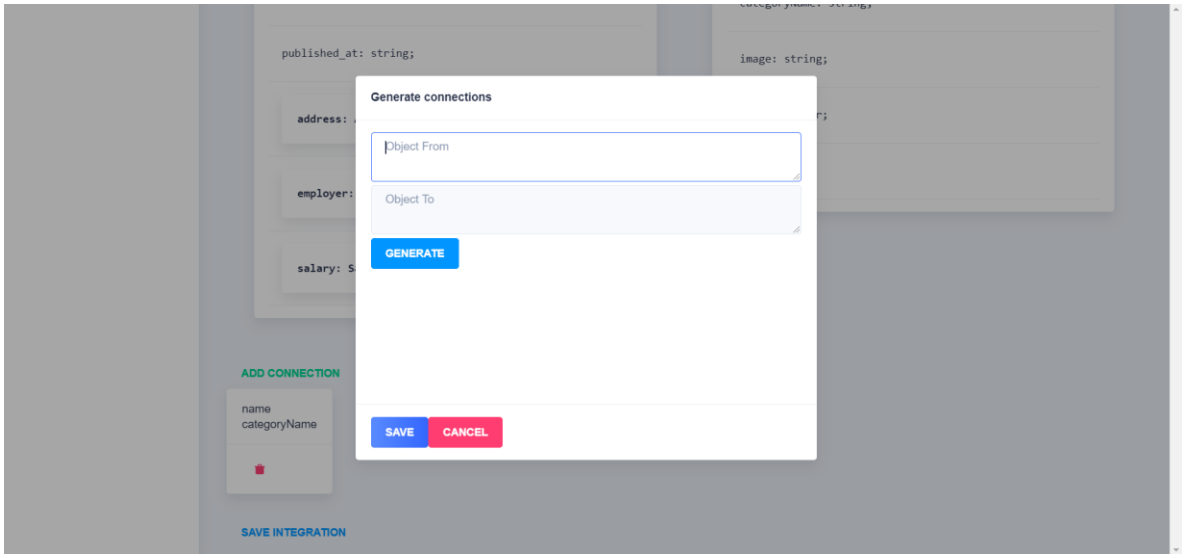


Рисунок 5.16 – Вікно генерування зв’язків

Object From – приклад об’єкту імпорту;

Object To - приклад об’єкту експорту з тими ж даними, що є в об’єкті імпорту, щоб система провела відповідність між полями.

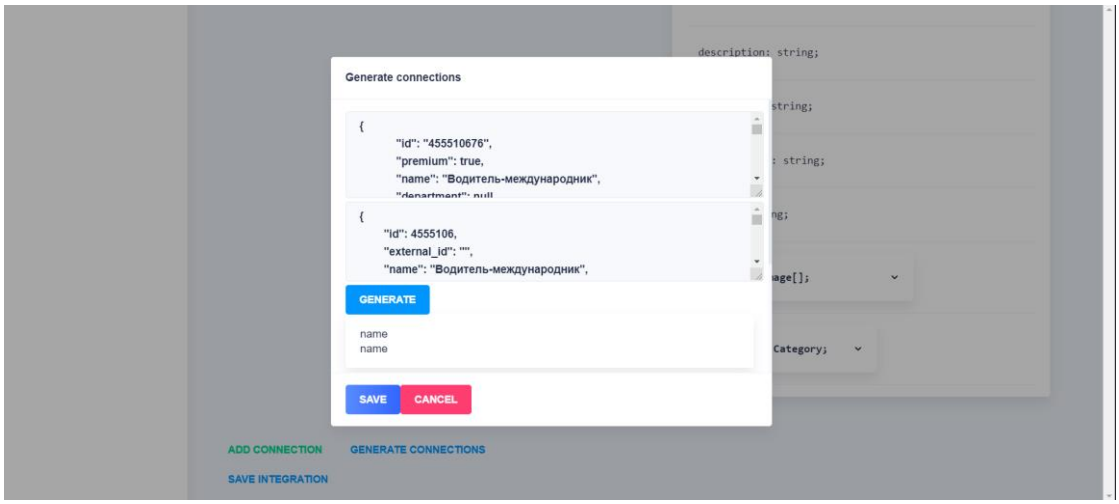


Рисунок 5.17 – Приклад заповнення полів генерування зв’язків

Після заповнення полів об’єктами необхідно натиснути кнопку «Generate» для генерування зв’язків.

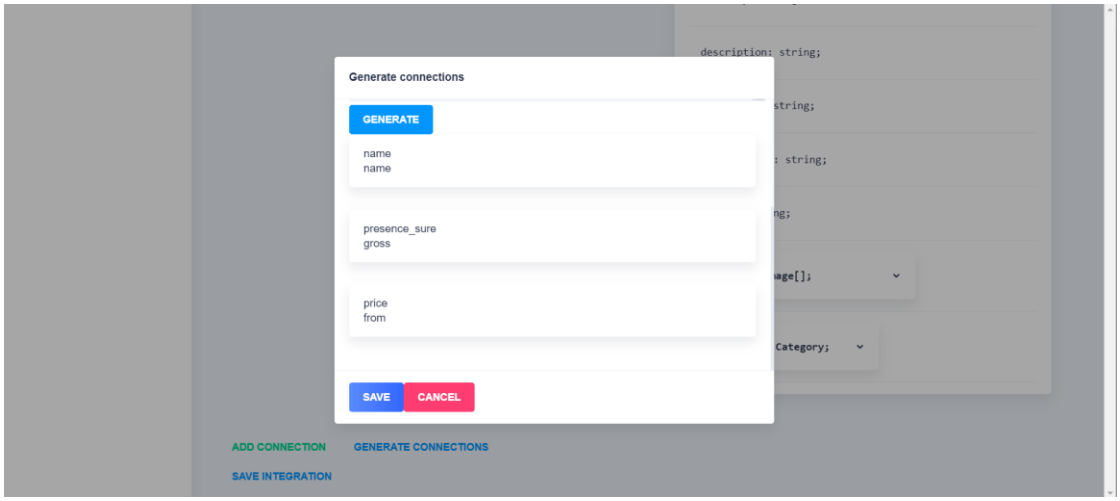


Рисунок 5.18 – Приклад результату генерування зв’язків

Щоб зберегти зв’язки необхідно натиснути кнопку «Save», щоб відмінити генерування – «Cancel».

Для збереження усієї інтеграції необхідно на сторінці створення інтеграції, що продемонстрована на Рисунках 5.12-5.15, натиснути кнопку «Save integration». Після збереження інтеграція буде відображена на сторінці інтеграцій, що показана на Рисунку 5.11. Для запуску інтеграції необхідно натиснути на зелений трикутник на картці інтеграції. Щоб переглянути результати інтеграції, тобто, транзакції – потрібно натиснути на синю іконку папки на картці інтеграції.

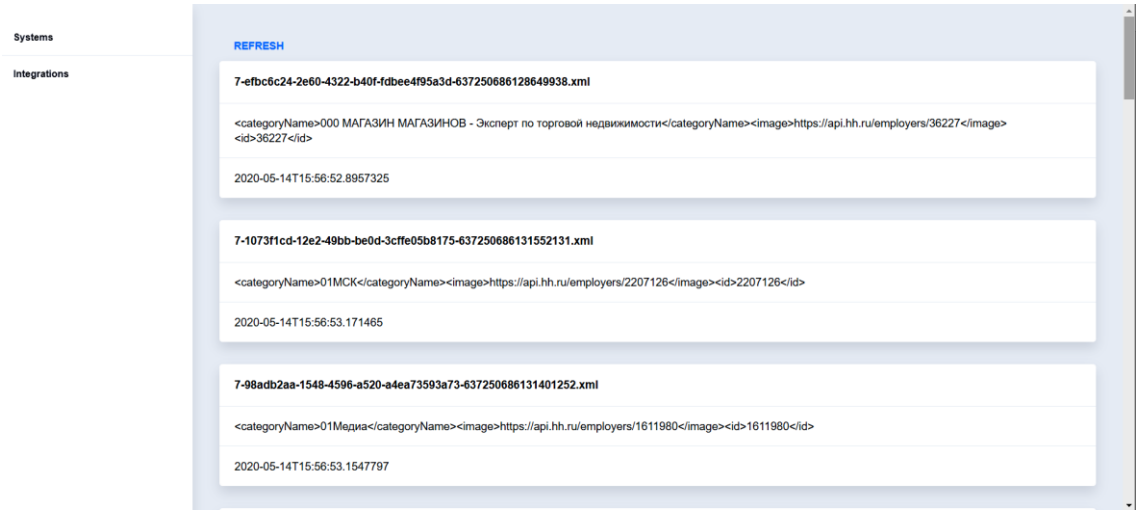


Рисунок 5.19 – Перегляд транзакцій інтеграції

Щоб оновити транзакції необхідно натиснути кнопку «Refresh». Також, створені об’єкти або файли можна побачити у системі експорту.

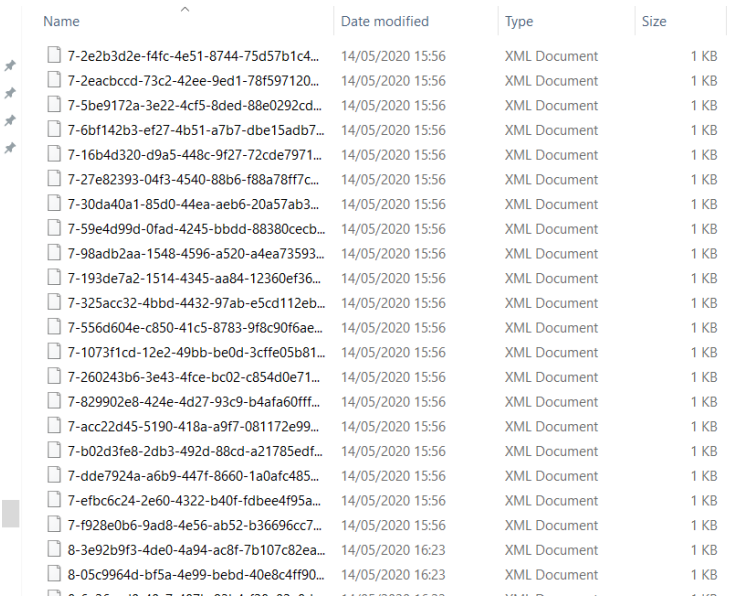


Рисунок 5.20 – Створені файли, що відповідають транзакціям

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Метою випробувань є перевірка відповідності функцій сервісу інтеграцій програмного забезпечення вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Для тестування системи було написано сценарії згідно з якими проходило тестування. Сценарії наведено у таблицях 5.1 – 5.6

Таблиця 5.1 – Трансформування об’єкту

Мета тесту	*Перевірка функції «Трансформування об’єкту»*
Початковий стан моделі	Завантажено новий об’єкт для трансформації
Вхідні дані	Вхідний об’єкт, опис його типів, типів вихідного об’єкту та зв’язки між ними
Схема проведення тесту	Відправка запиту до сервісу інтеграції на трансформацію з необхідними даними
Очікуваний результат	Отримано трансформований об’єкт вихідного типу
Стан моделі після проведення випробувань	Отримано трансформований об’єкт вихідного типу

Дане випробування було проведено з вхідними типами різної складності та вкладеності, всього написано 7 тестів для перевірки роботу алгоритму трансформування об’єктів.

Таблиця 5.2 – Генерування зв'язків між типами

Мета тесту	*Перевірка функції «Генерування зв'язків між типами»*
Початковий стан моделі	Завантажено об’єкти: вхідний та очікуваний результуючий, налаштовано специфікації типів
Вхідні дані	Вхідний об’єкт, опис його типів, типів вихідного об’єкту та вихідний об’єкт
Схема проведення тесту	Відправка запиту до сервісу інтеграції на генерування зв’язків з необхідними даними
Очікуваний результат	Згенеровано зв’язки за даними
Стан моделі після проведення випробувань	Згенеровано зв’язки за даними

Всього написано 2 тести для перевірки роботи алгоритму генерування зв’язків.

Таблиця 5.3 – Генерування типів

Мета тесту	*Перевірка функції «Генерування типів»*
Початковий стан моделі	Завантажено об’єкт
Вхідні дані	Вхідний об’єкт
Схема проведення тесту	Відправка запиту до сервісу інтеграції на генерування типів з необхідними даними

Очікуваний результат	Згенеровано специфікацію типів
Стан моделі після проведення випробувань	Згенеровано специфікацію типів

Таблиця 5.4 – Проведення інтеграції

Мета тесту	*Перевірка функції «Проведення інтеграції»*
Початковий стан моделі	Створено необхідні системи та їх типи
Вхідні дані	Значення параметрів доступу до систем, зв’язки між типами
Схема проведення тесту	Ввести вхідні дані, зберегти інтеграцію та натиснути на запуск інтеграції.
Очікуваний результат	Отримано транзакції з вихідними об’єктами.
Стан моделі після проведення випробувань	Отримано транзакції з вихідними об’єктами.

Таблиця 5.5 – Створення типу системи

Мета тесту	*Перевірка функції «Створення типу системи»*
Початковий стан моделі	Створено систему, відкрито типи цієї системи
Вхідні дані	Назва типу, поля типів (назва, тип поля, чи масив), закінчення url на дії з типом і обгортка об’єкту(ів), якщо необхідно.
Схема проведення тесту	Натиснуто кнопку “Create” на сторінці типів системи, введено

	необхідні дані та натиснуто кнопку “Save”.
Очікуваний результат	Створено новий тип системи.
Стан моделі після проведення випробувань	Створено новий тип системи.

Таблиця 5.6 – Створення системи

Мета тесту	*Перевірка функції «Створення системи»*
Початковий стан моделі	Відкрито вкладку з системами
Вхідні дані	Назва системи, тип доступу до системи, формат даних, url, закінчення url для різних дій з системою, якщо необхідно.
Схема проведення тесту	Натиснуто кнопку “Create” на сторінці систем, введено необхідні дані та натиснуто кнопку “Save”.
Очікуваний результат	Створено нову систему.
Стан моделі після проведення випробувань	Створено нову систему.

Висновок до розділу

У технологічному розділі було описано користувацьку інструкцію з експлуатації сервісу інтеграції програмного забезпечення.

Було проведене тестування функціоналу веб-застосунку, описані схеми проведення тестів і стани системи після випробувань.

.

ЗАГАЛЬНІ ВИСНОВКИ

У даній роботі реалізовано наступний проєкт – «Сервіс інтеграції програмного забезпечення». Була поставлена мета — збільшити можливості інтеграції програмного забезпечення завдяки створенню динамічних типів та систем, конфігурування точок доступу, налаштовуванню відношень між типами програмного забезпечення. Поставлені ціль та мета задачі були реалізовані.

Описано вхідні та вихідні дані та спроєктовано модель бази даних з 8 таблиць та 12 зовнішніх ключів.

У розділі математичного забезпечення були описані математичні задачі дипломного проєкту, розглянуте представлення типів як дерев, що з’єднуються у граф трансформації даних за допомогою відношень між типами, оглянуті різні варіанти вирішення поставлених задач, обґрунтовано вибір поточних рішень.

У розділі програмного та технічного забезпечення було розглянуто програми, мови програмування, каркаси за допомогою яких було реалізовано програмний продукт. Розглянуто загальні вимоги для технічних засобів, наведено специфікацію основних функцій системи з поясненням до них.

В ході виконання роботи, було досліджено інтеграцію систем з різним типом точок доступу, форматів та типів даних. Найшвидшими інтеграціями будуть ті, в яких вхідними і вихідними форматами даних є JSON та немає вкладених типів, це пов’язано з тим, що системі не потрібно проходити етапи зміни формату даних та заходити рекурсивно у внутрішні типи. Вкладені великі масиви також можуть сповільнити процес інтеграції.

Даний проєкт створений як науково–пізнавальний програмний продукт, який не призначений для великих масштабів використання. Він дозволяє лише ознайомитися із процесом і підходами до вирішення задач інтеграції програмного забезпечення.

ПЕРЕЛІК ПОСИЛАНЬ

1. David Chappell, "Enterprise Service Bus" c.10
2. Інтеграція | Енциклопедія Сучасної України [Електронний ресурс] – Режим доступу до ресурсу: http://esu.com.ua/search_articles.php?id=12384
3. ASP.NET Core Documentation | Microsoft Docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1
4. Angular [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/
5. James Rumbaugh, Ivar Jacobson, Grady Booch, «The unified modeling language reference manual» c.23
6. P. E. Hart, N. J. Nilsson, B. Raphael «A Formal Basis for the Heuristic Determination of Minimum Cost Paths» c. 100
7. Віндовз | Словопедія [Електронний ресурс] – Режим доступу до ресурсу: http://slovopedia.org.ua/29/53394/7937.html
8. Google Chrome [Електронний ресурс] – Режим доступу до ресурсу: https://googleblog.blogspot.com/2008/09/fresh-take-on-browser.html
9. C# | Microsoft Docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/previous-versions/aa336809(v=msdn.10)
10. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://web.archive.org/web/20060901003828/http://www.mozilla.org/js/

Додаток А

Тексти програмного коду

Сервіс інтеграції програмного забезпечення

(Найменування програми (документа))

DVD-R

(Вид носія даних)

10 арк, 244 Кб

(Обсяг програми (документа)

Київ – 2020 року

Файл IntegrationService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using AutoMapper;
using IntegrationCore.Data;
using IntegrationCore.Models.DB;
using IntegrationCore.Models.DTO;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using Newtonsoft.Json.Serialization;

namespace IntegrationCore.Services
{
    public class IntegrationService
    {
        private readonly IntegratorContext _context;
        private readonly HttpClient _mappingClient;
        private readonly IMapper _mapper;
        private readonly JsonSerializerSettings _jsonSerializerSettings = new JsonSerializerSettings
        {
            ContractResolver = new CamelCasePropertyNamesContractResolver()
```

```
};

public IntegrationService(IntegratorContext context, IMapper mapper,
    IHttpClientFactory clientFactory)
{
    _context = context;
    _mapper = mapper;
    _mappingClient = clientFactory.CreateClient("mapping-service");
}

public async Task RunIntegration(int id)
{
    var integration = await _context.Integration
        .Include("TypeFrom.SystemDefinition.ConnectionFields.ConnectionFieldValues")
        .Include("TypeTo.SystemDefinition.ConnectionFields.ConnectionFieldValues")
        .FirstAsync(el => el.Id == id);

    var fetchReqObj = new FetchRequest()
    {
        Type = integration.TypeFrom.SystemDefinition.TransferType,
        FetchType = new FetchData()
    };

    var filesFrom = new List<string>();

    fetchReqObj.FetchType.Action = ActionType.Get;
    fetchReqObj.FetchType.Path = integration.TypeFrom.SystemDefinition.Url +
integration.TypeFrom.UrlEnding + integration.TypeFrom.SystemDefinition.GetUrlEnding;
```

```
        fetchReqObj.FetchType.HeaderParams =
integration.TypeFrom.SystemDefinition.ConnectionFields
            .Where(el=>el.Role == FieldTransferRole.Header)
            .Select(el=> new FetchSimpleField()
            {
                Name = el.Name,
                Value = el.ConnectionFieldValues.First(el=>el.IntegrationId ==
integration.Id).Value
            }).ToList();
        fetchReqObj.FetchType.QueryParams =
integration.TypeFrom.SystemDefinition.ConnectionFields
            .Where(el => el.Role == FieldTransferRole.Query)
            .Select(el => new FetchSimpleField()
            {
                Name = el.Name,
                Value = el.ConnectionFieldValues.First(el => el.IntegrationId ==
integration.Id).Value
            }).ToList();


        var fetchContent = new StringContent(JsonConvert.SerializeObject(fetchReqObj,
_jsonSerializerSettings), Encoding.UTF8, "application/json");
        var fetchResult = await _mappingClient.PostAsync("fetch-data", fetchContent);
        filesFrom.Add(await fetchResult.Content.ReadAsStringAsync());


        if (integration.TypeFrom.SystemDefinition.DataType == DataType.Xml)
        {
            for(var i = 0; i< filesFrom.Count; i++)
            {
```

```
                filesFrom[i] = filesFrom[i].Replace("\"", "\"");
            }
        }
        var parsedData = new List<string>();

        foreach (var file in filesFrom)
        {
            var req = new
            {
                type = ((int)integration.TypeFrom.SystemDefinition.DataType).ToString(),
                data = file
            };
            var parseContent = new StringContent(JsonConvert.SerializeObject(req,
_jsonSerializerSettings), Encoding.UTF8, "application/json");
            var parseResult = await _mappingClient.PostAsync("parse-data", parseContent);
            parsedData.Add(await parseResult.Content.ReadAsStringAsync());
        }

        if (!string.IsNullOrEmpty(integration.TypeFrom.GetFieldWrapper))
        {
            var newParsedData = new List<string>();
            foreach (var item in parsedData)
            {
                var cont = JObject.Parse(item);
                if (cont.ContainsKey(integration.TypeFrom.GetFieldWrapper))
                {
                    var values =
cont.GetValue(integration.TypeFrom.GetFieldWrapper).ToObject<IEnumerable<dynamic>>();
```

```
            var arrayValues = values.Select(el=> (string) JsonConvert.SerializeObject(el));

            newParsedData
                .AddRange(arrayValues);
        }
    }

    parsedData = newParsedData;
}

var mappedData = new List<string>();
var types = await _context.TypeDefinition
    .Where(el => el.SystemId == integration.TypeFrom.SystemId ||
                el.SystemId == integration.TypeTo.SystemId || el.IsBasic)
    .Include(el=>el.Fields).ToListAsync();

var connections = await _context
    .FieldConnection.Where(el => el.IntegrationId == integration.Id)
    .ToListAsync();

foreach (var item in parsedData)
{
    var req = new
    {
        objFrom = JsonConvert.DeserializeObject(item),
        types = _mapper.Map<IEnumerable<TypeDefinitionDto>>(types),
        typeFromId = integration.TypeFromId,
```

```
            typeToId = integration.TypeToId,
            connections =
_mapper.Map<IEnumerable<FlatFieldConnectionDto>>(connections)
        };

        var mapContent = new StringContent(JsonConvert.SerializeObject(req,
_jsonSerializerSettings), Encoding.UTF8, "application/json");
        var mapResult = await _mappingClient.PostAsync("map-data", mapContent);
        mappedData.Add(await mapResult.Content.ReadAsStringAsync());
    }

    var writeData = new List<string>();
    foreach (var item in mappedData)
    {
        var req = new
        {
            data = JsonConvert.DeserializeObject(item),
            type = integration.TypeTo.SystemDefinition.DataType
        };

        var writeContent = new StringContent(JsonConvert.SerializeObject(req,
_jsonSerializerSettings), Encoding.UTF8, "application/json");
        var writeResult = await _mappingClient.PostAsync("write-data", writeContent);
        writeData.Add(await writeResult.Content.ReadAsStringAsync());
    }

    var postReqObj = new FetchRequest()
    {
        Type = integration.TypeTo.SystemDefinition.TransferType,
```

```
            FetchType = new FetchData(),
        };


        postReqObj.FetchType.Action = ActionType.Create;
        postReqObj.FetchType.Path = integration.TypeTo.SystemDefinition.Url +
integration.TypeTo.UrlEnding + integration.TypeTo.SystemDefinition.PostUrlEnding;
        postReqObj.FetchType.HeaderParams =
integration.TypeTo.SystemDefinition.ConnectionFields
            .Where(el => el.Role == FieldTransferRole.Header)
            .Select(el => new FetchSimpleField()
            {
                Name = el.Name,
                Value = el.ConnectionFieldValues.FirstOrDefault(el => el.IntegrationId ==
integration.Id).Value
            })
            .Where(el=>el.Value!= null)
            .ToList();
        postReqObj.FetchType.QueryParams =
integration.TypeTo.SystemDefinition.ConnectionFields
            .Where(el => el.Role == FieldTransferRole.Query)
            .Select(el => new FetchSimpleField()
            {
                Name = el.Name,
                Value = el.ConnectionFieldValues.FirstOrDefault(el => el.IntegrationId ==
integration.Id).Value
            })
            .Where(el => el.Value != null)
            .ToList();
```

```
        foreach (var item in writeData)
        {
            postReqObj.Data = item;
            var name = $"{integration.Id}-{Guid.NewGuid()}-
{DateTime.Now.Ticks}.{integration.TypeTo.SystemDefinition.DataType.ToString().ToLower()
}";
            if (postReqObj.Type == TransferType.Ftp)
            {
                postReqObj.FetchType.QueryParams.Add(new FetchSimpleField()
                {
                    Name = "name",
                    Value = name
                });
            }

            var postContent = new StringContent(JsonConvert.SerializeObject(postReqObj,
_jsonSerializerSettings), Encoding.UTF8, "application/json");
            var postResult = await _mappingClient.PostAsync("fetch-data", postContent);
            var result = await postResult.Content.ReadAsStringAsync();

            if (postReqObj.Type == TransferType.Ftp)
            {

postReqObj.FetchType.QueryParams.Remove(postReqObj.FetchType.QueryParams.First(el=>el
.Value == name));
            }

            await _context.Transaction.AddAsync(new Transaction()
            {
```

```
                Content = item,
                GivenName = name,
                IntegrationId = integration.Id,
                Timestamp = DateTime.Now
            });
        }

        await _context.SaveChangesAsync();
    }
  }
}
```

Файл GeneratorService.cs

```
using AutoMapper;
using IntegrationCore.Data;
using IntegrationCore.Models.DB;
using IntegrationCore.Models.DTO;
using Microsoft.EntityFrameworkCore;
using Newtonsoft.Json;
using Newtonsoft.Json.Serialization;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
```

```
namespace IntegrationCore.Services
{
    public class GeneratorService
    {
        private readonly IntegratorContext _context;
        private readonly HttpClient _mappingClient;
        private readonly IMapper _mapper;
        private readonly JsonSerializerSettings _jsonSerializerSettings = new JsonSerializerSettings
        {
            ContractResolver = new CamelCasePropertyNamesContractResolver()
        };

        public GeneratorService(IntegratorContext context, IMapper mapper,
            IHttpClientFactory clientFactory)
        {
            _context = context;
            _mapper = mapper;
            _mappingClient = clientFactory.CreateClient("mapping-service");
        }

        public async Task<IEnumerable<TypeDefinitionDto>> GenerateTypes(int systemId, string
data, string name, bool addDefaultValues)
        {
            var system = await _context.SystemDefinition.FindAsync(systemId);
            var types = await _context.TypeDefinition
                .Where(el => el.IsBasic)
                .Include(el => el.Fields).ToListAsync();
```

```
            var genReq = new
            {
                data = await TransformObject(data, system.DataType),
                name,
                types = _mapper.Map<IEnumerable<TypeDefinitionDto>>(types),
                addDefaultValues
            };
            var generateContent = new StringContent(JsonConvert.SerializeObject(genReq,
_jsonSerializerSettings), Encoding.UTF8, "application/json");
            var generateResult = await _mappingClient.PostAsync("type-generator",
generateContent);

            return JsonConvert.DeserializeObject<IEnumerable<TypeDefinitionDto>>(
                await generateResult.Content.ReadAsStringAsync());
        }

        public async Task<IEnumerable<FieldConnection>> GenerateConnections(int typeFromId,
int typeToId, string objFrom, string objTo)
        {
            var typeFrom = await _context.TypeDefinition
                .Include(el=>el.SystemDefinition)
                .FirstOrDefaultAsync(el=>el.Id == typeFromId);

            var typeTo = await _context.TypeDefinition
                .Include(el => el.SystemDefinition)
                .FirstOrDefaultAsync(el => el.Id == typeToId);

            var types = await _context.TypeDefinition
                .Where(el => el.SystemId == typeFrom.SystemId ||
```

```
                el.SystemId == typeTo.SystemId || el.IsBasic)
            .Include(el => el.Fields).ToListAsync();


        var genReq = new
        {
            objFrom = await TransformObject(objFrom, typeFrom.SystemDefinition.DataType),
            objTo = await TransformObject(objTo, typeTo.SystemDefinition.DataType),
            types = _mapper.Map<IEnumerable<TypeDefinitionDto>>(types),
            typeFromId,
            typeToId
        };
        var generateContent = new StringContent(JsonConvert.SerializeObject(genReq,
_jsonSerializerSettings), Encoding.UTF8, "application/json");
        var generateResult = await _mappingClient.PostAsync("connections-generator",
generateContent);
        return JsonConvert.DeserializeObject<ConnectionGenerationResult>(
            await generateResult.Content.ReadAsStringAsync()).Connections;
    }
    private async Task<object> TransformObject(string obj, DataType dataType)
    {
        if (dataType == DataType.Xml)
        {
            obj = obj.Replace("\"", "\'");
        }
        var req = new
        {
            type = ((int)dataType).ToString(),
            data = obj
```

```
            };

            var parseContent = new StringContent(JsonConvert.SerializeObject(req,
_jsonSerializerSettings), Encoding.UTF8, "application/json");

            var parseResult = await _mappingClient.PostAsync("parse-data", parseContent);

            return JsonConvert.DeserializeObject(await parseResult.Content.ReadAsStringAsync());

        }

    }
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проєкту	**В.о. завідувача кафедри**
_____ *Юрій ОЛІЙНИК*	_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)	(підпис) (вл. ім’я, прізвище)
“13” квітня 2020 р.	“14” квітня 2020 р.

Сервіс інтеграції програмного забезпечення

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6229.01.000.ТЗ*

на 10 сторінках

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ…….. 3

1.1 Повне найменування системи та її умовне позначення…….. 3

1.2 Найменування організації-замовника та організацій-учасників робіт…… 3

1.3 Перелік документів, на підставі яких створюється система (Завдання ДП)……... на 3

1.4 Планові терміни початку і закінчення роботи зі створення системи…….. 4

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СЕРВІСУ……. 5

2.1 Призначення сервісу…….. 5

2.2 Цілі створення сервісу…….. 5

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ…… 6

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ……..... 7

4.1 Вимоги до функціональних характеристик…….. 7

4.2 Вимоги до надійності…….. 7

4.3 Умови експлуатації (тільки для систем, специфіка яких передбачає особливі умови експлуатації)…… 7

4.4 Вимоги до складу і параметрів технічних засобів…… 7

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ…… 9

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ……. 10

6.1 Види випробувань…… 10

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: Сервіс інтеграції програмного забезпечення

Коротке найменування системи: «Integration Core».

1.2 Найменування організації-замовника та організації-учасника робіт

Замовником є кафедра автоматизованих систем обробки інформації та управління Національного технічного університету України "Київський політехнічний інститут ім. Ігоря Сікорського" (далі за текстом — Замовник). Адреса замовника: м. Київ, п. Перемоги 37, 18 корпус ФІОТ.

Розробник сервісу — студентка групи ІС-з61 кафедри автоматизованих систем обробки інформації та управління Національного технічного університету України "Київський політехнічний інститут ім. Ігоря Сікорського" Щербакова Анастасія Дмитрівна.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням сервісу інтеграцій – 1 грудня 2019 року.

Плановий термін по закінченню роботи над створенням системи інтеграцій – 22 травня 2020 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Метою розробки є збільшення можливостей інтеграції програмного забезпечення завдяки створенню динамічних типів та систем, конфігурування точок доступу, налаштовуванню відношень між типами програмного забезпечення.

2.2 Цілі та задачі створення системи

Основним показник ефективності який збільшує дана інформаційна система – це проведення та налаштовування інтеграцій між системи за короткий час без потреби у написанні коду. Ефективність інтеграцій підприємств, що будуть використовувати сервіс інтеграції однозначно збільшиться, адже голова проектів буде мати повну звітність щодо виконаної роботи і не буди мати потреби у створенні нових програмних забезпечень, що напрямлені на інтеграцію специфічних систем.

Ціль системи – розширити можливості інтеграцій для підприємств та покращити показник швидкості введення інтеграцій у існуючу інфраструктуру підприємства.

Задачі системи:

- завантаження даних з різних джерел за допомогою налаштування параметрів доступу;
- трансформування даних за схемами типів, заданими користувачем;
- експорт даних у поточну систему;
- збереження результатів транзакцій.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Для того, щоб користуватися системою, користувач повинен зайти на сайт сервісу інтеграцій.

Об'єктом автоматизації є процеси інтеграції між системами на підприємстві з динамічним налаштуванням трансформації типів.

Для того, щоб вирішити задачу трансформування типів під час інтеграцій, необхідно пройти такі етапи:

- динамічне конфігурування типів і ведення даних для доступу до системи;
- імпорт даних з системи, що є джерелом даних;
- трансформування даних за конфігурацією типів;
- експорт даних у поточну систему;

Для того, щоб автоматизувати цей процес, необхідно розв'язати задачу пошуку оптимального шляху під час трансформування даних.

По завершенню роботи ми отримаємо систему, яка дозволить автоматизувати процес інтеграції програмного забезпечення.

4. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Задача підготовки побудови системи інтеграції програмного забезпечення вимагає наявних даних про типи та їх зв’язки та алгоритму знаходження шляху від внутрішніх елементів до типу-кореня.

Задача підготовки типів та зв’язків вимагає визначення можливих типів системи для інтеграції.

4.2 Вимоги до надійності

Система повинна адекватно реагувати на помилки застосування та видавати відповідні повідомленні користувачеві, а також записувати їх у log-файл.

Система повинна знаходити найоптимальніший шлях від базових з’єднаних типів до коренів типів що інтегруються. Після знаходження шляху сервіс повинен перенести дані з поля базового типу системи-імпорту у поле базового типу системи-експорту.

4.3 Умови експлуатації

Для адекватної роботи системи необхідний пристрій з платформою, яка відповідає вимогам зазначеним в розділі 4.4 .

Усі користувачі системи повинні дотримуватися правил експлуатації електронної обчислювальної техніки.

4.4 Вимоги до складу і параметрів технічних засобів

Даний програмний продукт представлений у вигляді веб додатку і складається тільки з трьох частин: клієнтська, серверна, та окремий сервіс, що виконує інтеграції.

Для коректної роботи сайту потрібно мати один із таких браузерів з відповідною версією:

– Safari 12.0+;

– Firefox 66.0+;

– Google Chrome 74.0+;

– Chromium 71+.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

У таблиці 5.1 наведено календарний план робіт та терміни їх виконання.

Таблиця 5.1 – Календарний план виконання робіт

№ з/п	Назва етапів створення продукту	Строк виконання
1.	Вивчення рекомендованої літератури	15.02.2020 р.
2.	Аналіз існуючих методів розв’язання задачі	22.02.2020 р.
3.	Постановка та формалізація задачі	01.03.2020 р.
4.	Розробка інформаційного забезпечення	15.03.2020 р.
5.	Алгоритмізація задачі	22.03.2020 р.
6.	Обґрунтування використовуваних технічних засобів	30.03.2020 р.
7.	Розробка програмного забезпечення	20.04.2020 р.
8.	Налагодження програми	27.04.2020 р.
9.	Виконання графічних документів	04.05.2020 р.
10.	Оформлення пояснювальної записки	18.05.2020 р.
11.	Подання ДП на попередній захист	22.05.2020 р.
12.	Подання ДП на основний захист	01.05.2020 р.
13.	Подання ДП рецензенту	05.06.2020 р.

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Види випробувань узгоджуються із замовником до проведення випробувань. Здача - прийом робіт виконується поетапно на комп'ютерах замовника в аудиторіях кафедри АСОІУ у відповідності з робочою програмою та календарним планом.

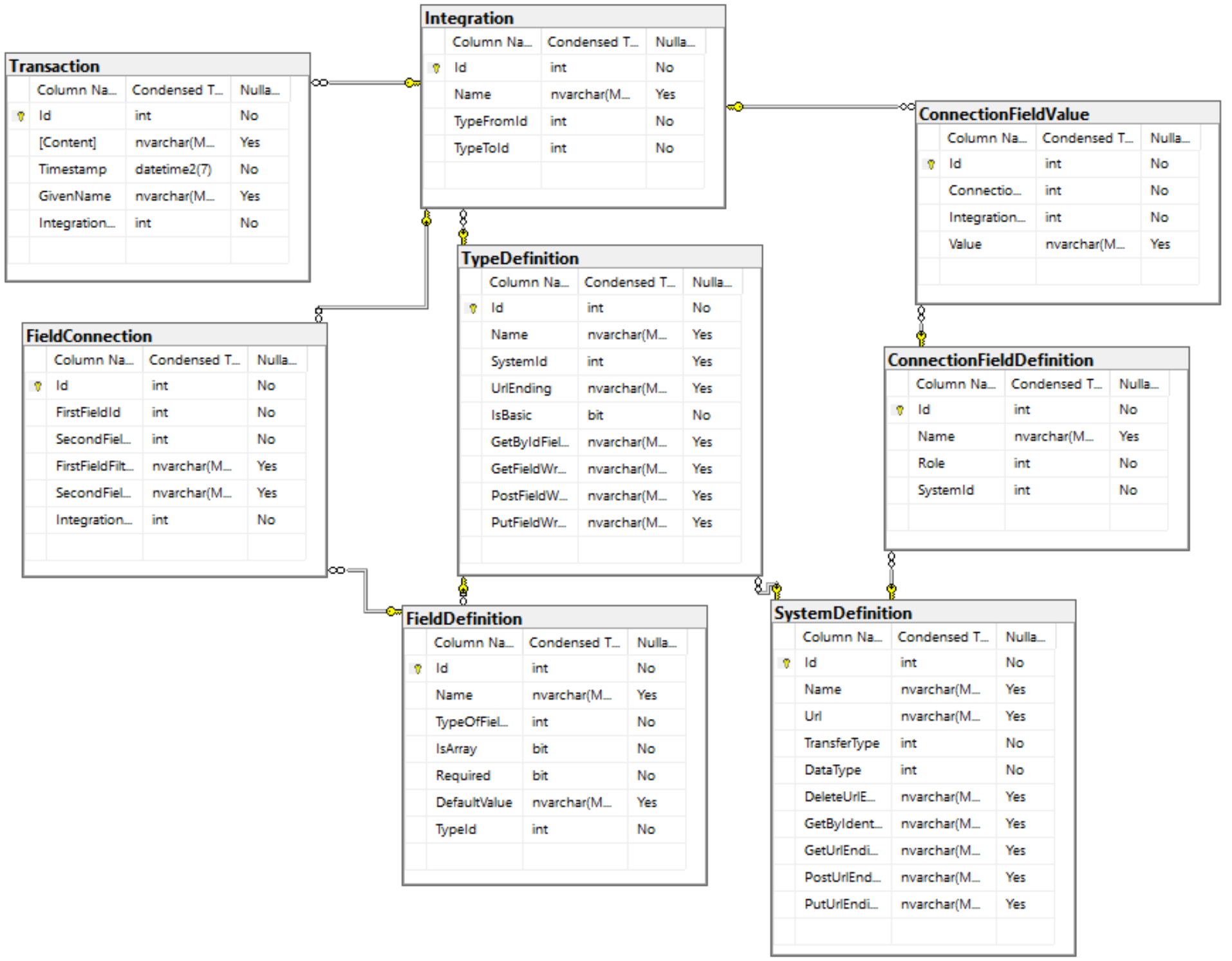
Усі програмні продукти, що створюються в рамках даної системи передаються замовнику як у вигляді готових модулів, так і у вигляді вихідних кодів, представлених в електронній формі.

Графічний матеріал
до дипломного проєкту

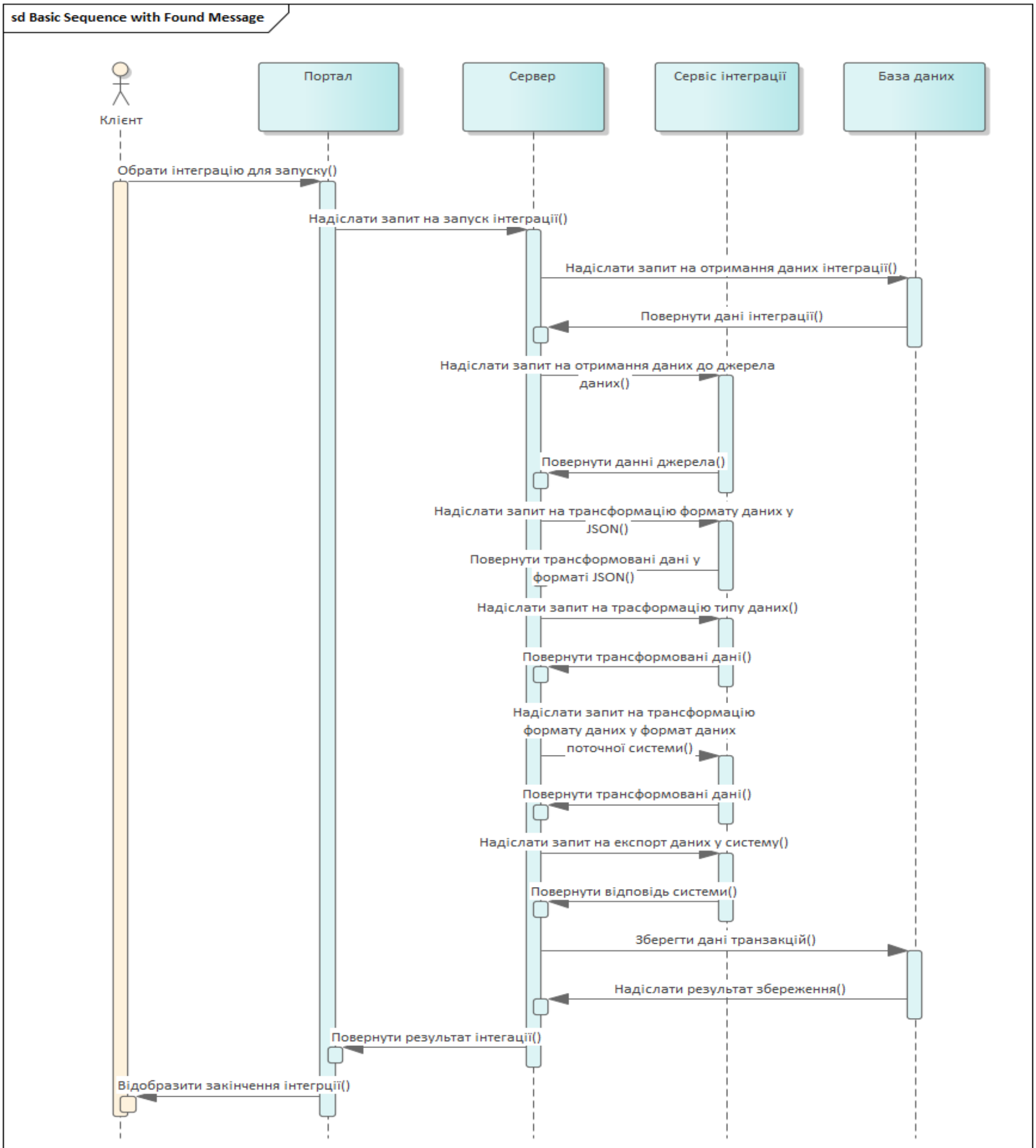
на тему: Сервіс інтеграції програмного забезпечення

Київ – 2020 року

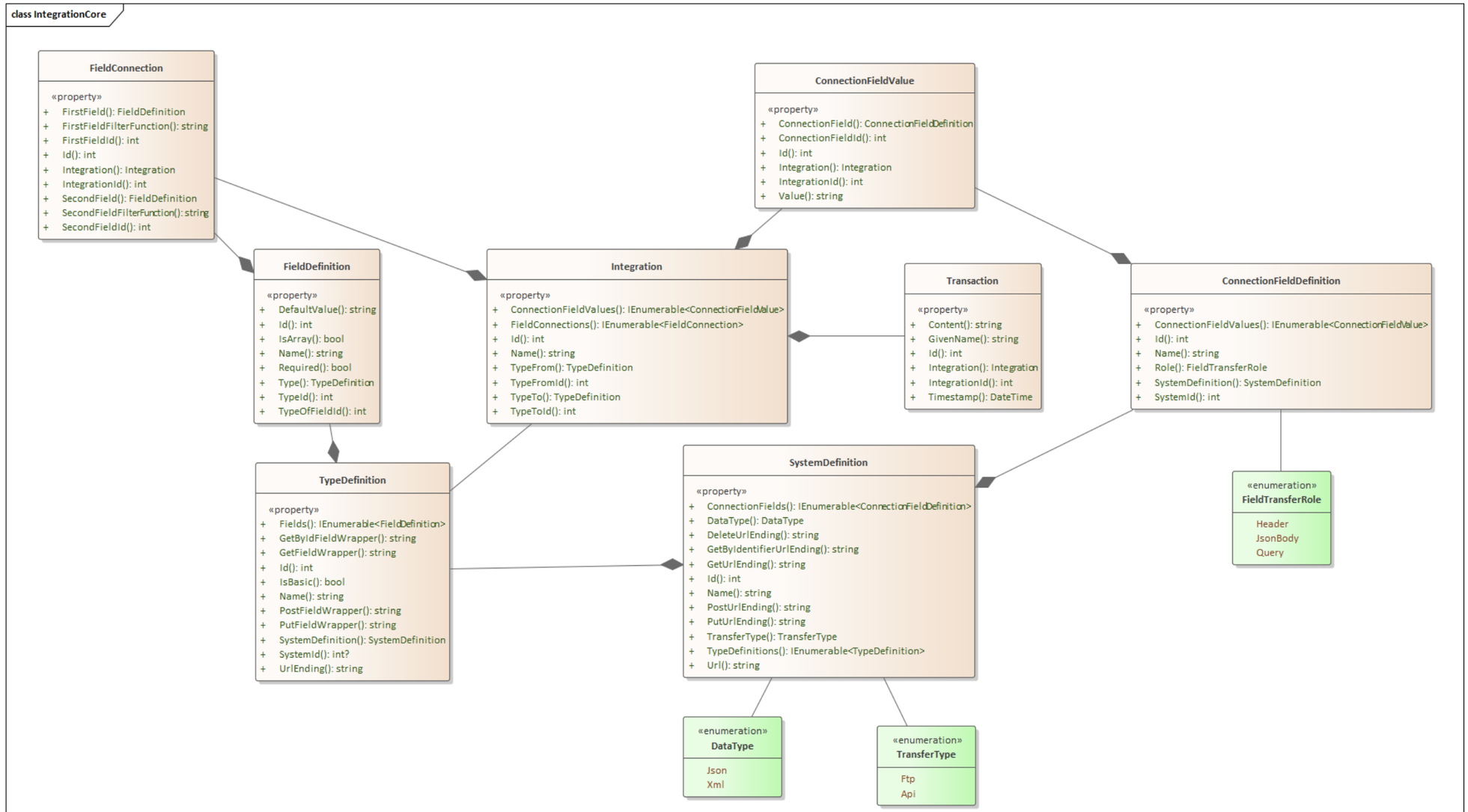
ДП 6229.02.000.СБД



Зм.	Арк.	№ докум.	Підп.	Дата	ДП 6229.02.000.СБД		
					Схема бази даних	Лит.	Маса / Масштаб
Розроб.		Щербакова А.Д.					
Перев.		Олійник Ю.О.				Аркуш 1	Аркушів 6
Т. Кон.					Сервіс інтеграції програмного забезпечення	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-з61	
Н. Кон.		Тєлишева Т.О.					
Затв.		Олійник Ю.О.					

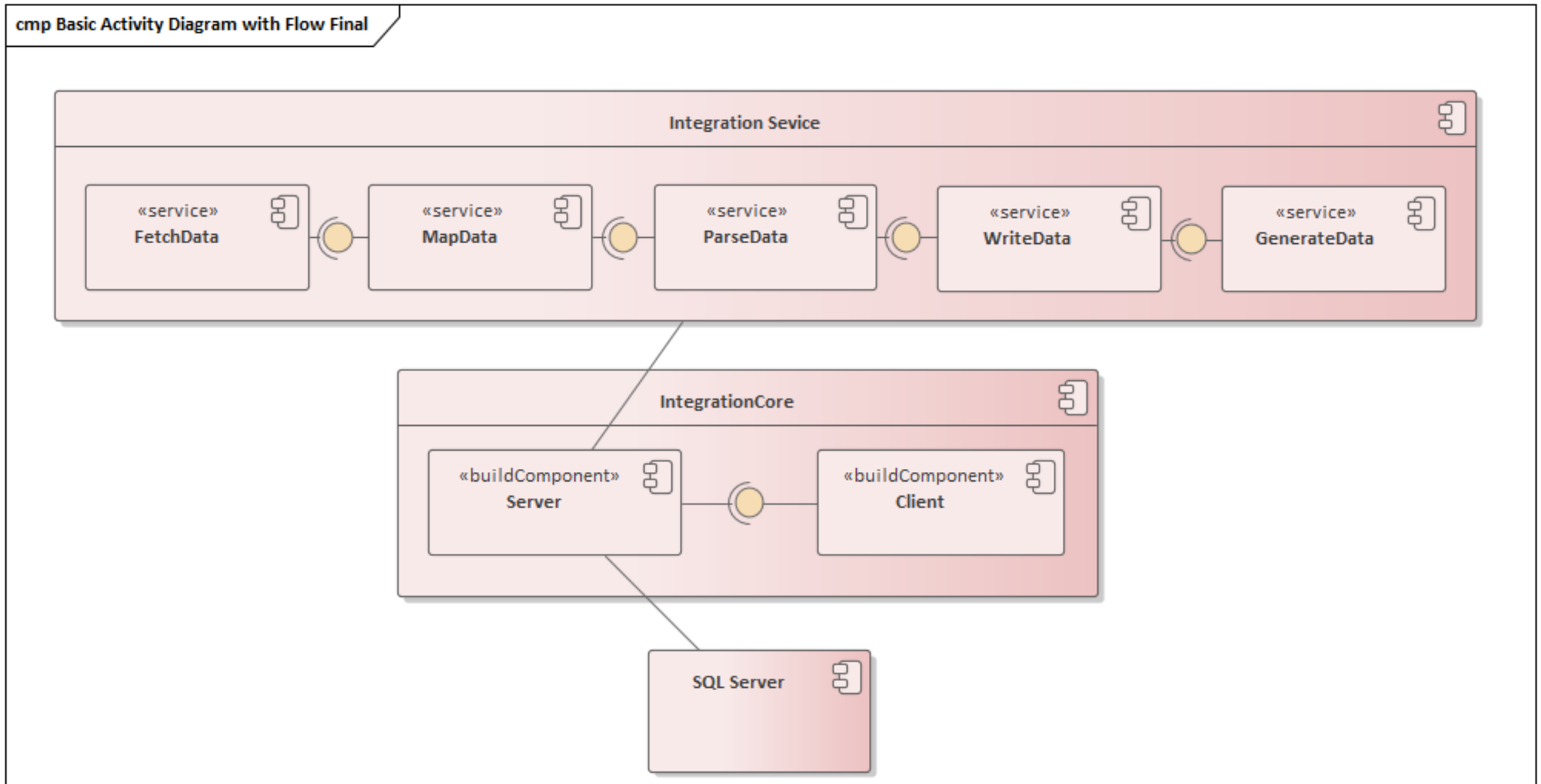


Зм.	Арк.	№ докум.	Підп.	Дата	ДП 6229.03.000.ССП		
Розроб.		Щербакова А.Д.			Схема структурна послідовності	Лит.	Маса / Масштаб
Перев.		Олійник Ю.О.				Аркуш 2	Аркушів 6
Т. Кон.					Сервіс інтеграції програмного забезпечення	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-з61	
Н. Кон.		Тєлишева Т.О.					
Затв.		Олійник Ю.О.					

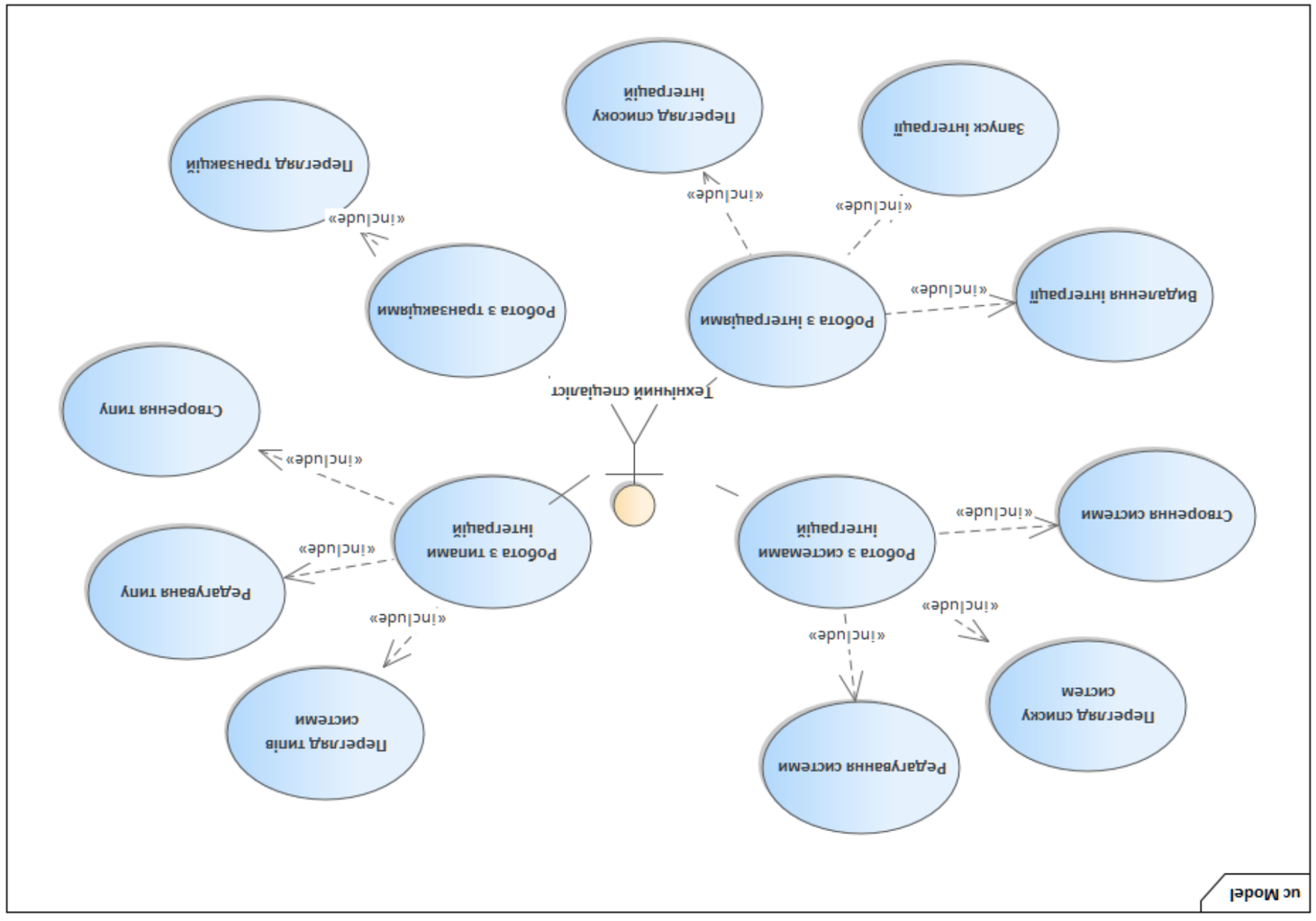


Зм.	Арк.	№ докум.	Підп.	Дата	ДП 6229.04.000.ССК		
Розроб.		Щербакова А.Д.			Схема структурна класів програмного забезпечення	Лит.	Маса	Масштаб
Перев.		Олійник Ю.О.						
Т. Кон.						Аркуш 3	Аркушів 6	
Н. Кон.		Телишева Т.О.			Сервіс інтеграції програмного забезпечення	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-з61		
Затв.		Олійник Ю.О.						

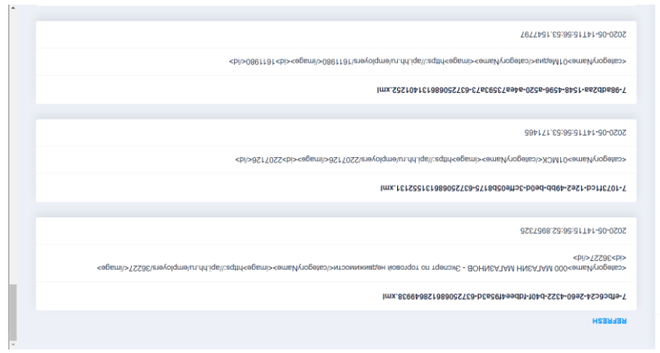
ДП 6229.05.000.ССК

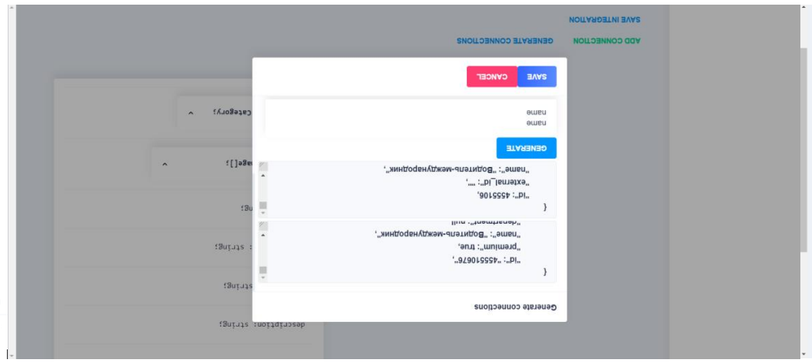


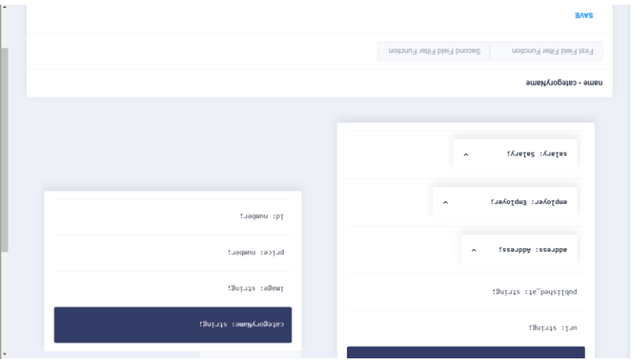
Зм.	Арк.	№ докум.	Підп.	Дата	ДП 6229.05.000.ССК		
Розроб.		Щербакова А.Д.			Схема структурна компонентів програмного	Лит.	Маса	Масштаб
Перев.		Олійник Ю.О.						
Т. Кон.						Аркуш 4	Аркушів 6	
Н. Кон.		Тєлишева Т.О.			Сервіс інтеграції програмного забезпечення	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-з61		
Затв.		Олійник Ю.О.						

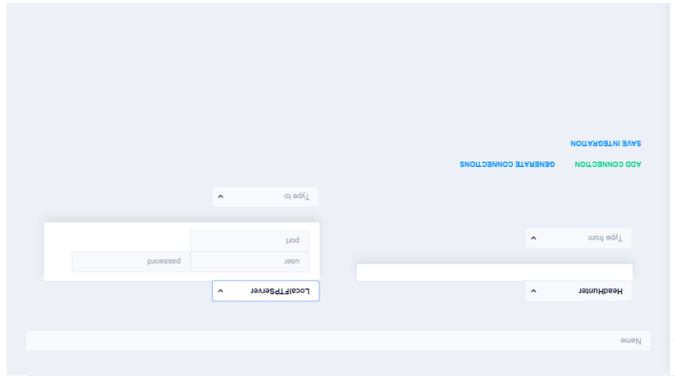


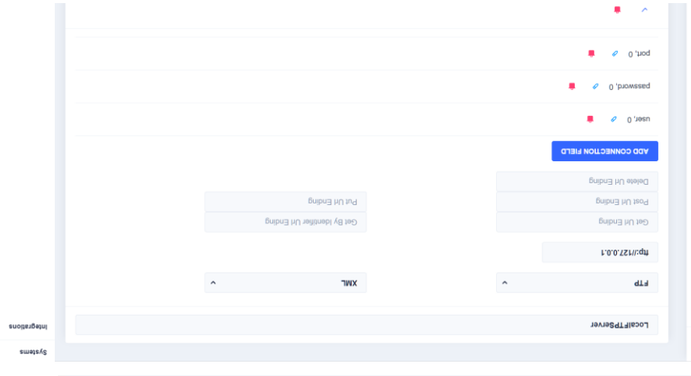
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*	ДП 6229.06.000.ССВ		
Розроб.		Шербакова А.Л.				*Літ.*	*Маса*	*Масштаб*
Перев.		Олійник Ю.О.			Схема структура варіантів використань			
Т. Кон.						*Арк.* 5	*Аркушів* 6
Н. Кон.		Тесницева Т.О.			Серія інтерфейсів програмного забезпечення	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-361	
Затв.		Олійник Ю.О.					

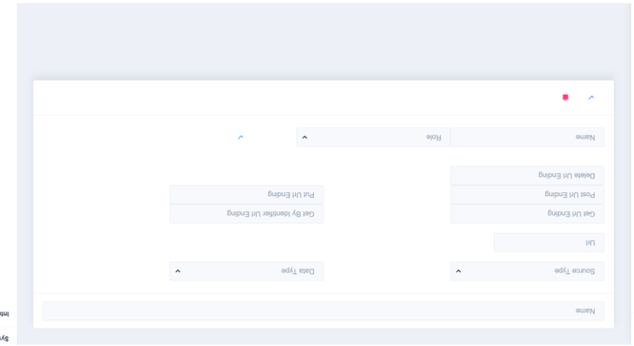


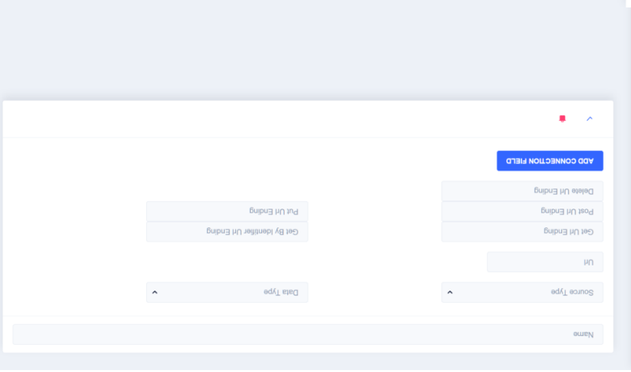


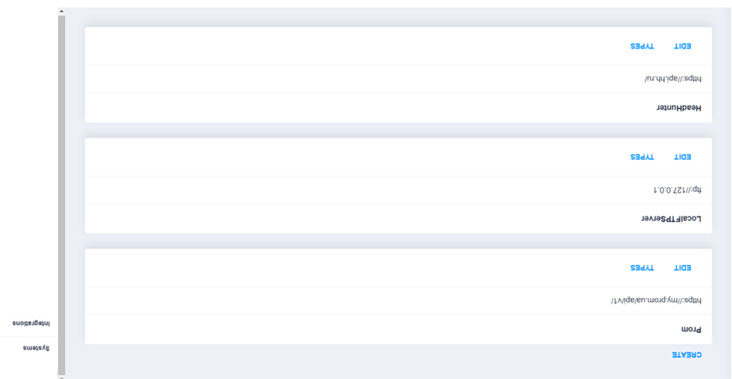


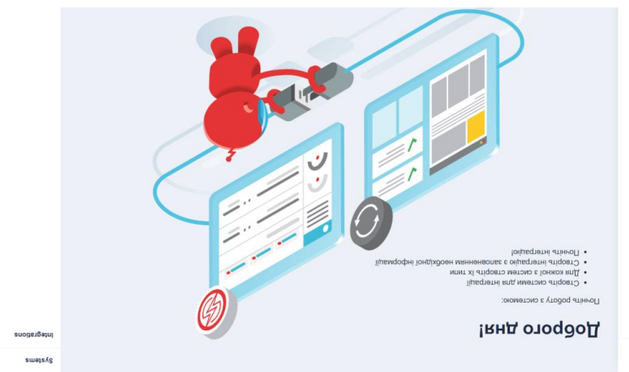












Зм.	Арк.	№ докум.	Підп.	Дата
Розроб.	Шевцова А.Л.			
Перев.	Олійник Ю.О.			
Т. Кон.				
Н. Кон.	Тесленова Т.О.			
Затв.	Олійник Ю.О.			

ДП 6229.07.000.КЕ

Крос-платформний додаток для керування форм

Літ.	Маса	Масштаб

Аркуш 6 Аркушів 6

Сервіс інтеграцій програмного забезпечення

КПІ ім. Ігоря Сікорського
кафедра АСОІУ гр. ІС-361

ДП 6229.07.000.КЕ